CoSelTune: Co-Designing Token Selection and Backward Computation for Efficient Long-Context Fine-Tuning

Anonymous authors
Paper under double-blind review

Abstract

Fine-tuning adapts large language models to downstream tasks but incurs high memory and computational costs, especially with long documents. Parameter-efficient methods can reduce trainable parameters. While token-selective methods such as TokenTune and TokenSeek further reduce activation storage by restricting backpropagation to selected tokens, substantial memory and computational costs still remain. We propose CoSelTune, a modular fine-tuning method that co-designs lightweight token selection and backward computation. It reserves the question-and-answer segment following the document (the suffix) and selects document positions by relevance and coverage within a fixed backward budget. Since suffix queries require the full document, CoSelTune separates their scans from those of selected document queries and compresses the projection inputs needed for local adapter updates, retaining full forward context and all answer losses. CoSelTune can be combined with existing efficient fine-tuning methods for additional resource savings. Experiments on question answering and summarization show memory savings and faster training with comparable performance.

1 Introduction

Fine-tuning adapts a pretrained language model using task- or domain-specific data (Howard & Ruder, 2018). It helps models follow instructions and generate responses suited to downstream tasks (Ouyang et al., 2022). Full fine-tuning of large language models (LLMs) updates all parameters, incurring memory costs for gradients, optimizer states and separate adapted model copies (Hu et al., 2022; Houlsby et al., 2019).

To reduce these adaptation costs, parameter-efficient fine-tuning (PEFT) trains a small parameter set with a largely frozen backbone. Adapters insert trainable modules (Houlsby et al., 2019), prefix tuning learns continuous task-specific prefixes (Li & Liang, 2021), and LoRA represents weight updates with low-rank factors (Hu et al., 2022). QLoRA combines low-rank adapters with a quantized backbone (Dettmers et al., 2023). Activation storage and attention backpropagation remain major costs when training on long documents (Simoulin et al., 2024; Chen et al., 2024).

These costs motivate changes to attention and backpropagation. LongLoRA uses shifted sparse attention during context extension (Chen et al., 2024), while SeCO and SpaCO organize backpropagation by chunks (Li et al., 2025). TokenTune samples backward positions uniformly (Simoulin et al., 2024), and TokenSeek uses attention and activation-gradient scores for instance-aware selection (Zeng et al., 2026). Both retain full-context forward computation. Under our whole-sequence budget, their selection can omit supervised prediction positions in the question-and-answer suffix along with document states. TokenSeek also incurs model-based scoring cost. Even a small selected set can require long scans over attention keys if it includes queries near the end of the document. The selected positions therefore affect both supervision coverage and execution cost.

The document supplies context, while the question-and-answer suffix contains the supervised prediction positions. CoSelTune reserves the complete suffix within a fixed backward budget and selects document positions for question relevance and coverage. Preserving the suffix also keeps a common scan boundary at the sequence end, so document selection alone cannot shorten every query's scan. We separate query work over shared key/value (K/V) tensors. Document queries scan through the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

last selected document position, while suffix queries keep access to the complete causal context. Unselected document positions can still contribute to local K/V adapter updates, whose required inputs are compressed. Shared storage and recomputation reduce the remaining activation costs. The full forward input is retained, and the method works with LoRA or QLoRA.

On Qwen3-14B, CoSelTune reduces total training time versus TokenSeek without gradient checkpointing by 32.4% with LoRA and 30.8% with QLoRA. Separate 64K tests peak at 59.59 and 41.33 GiB, respectively. Its five-task average exceeds TokenTune and TokenSeek respectively by 1.35 and 1.63 points with LoRA and 0.40 and 1.49 points with QLoRA. Our contributions are:

- A token-allocation strategy that preserves the complete supervised suffix and selects document positions by relevance and coverage under a shared budget.
- A joint execution design that gives document and suffix queries separate scan ranges over shared K/V, preserves their local adapter contributions, and compresses the projection inputs those updates require.
- Five-task evaluations of LoRA on two model families and QLoRA on Qwen3, with resource measurements and controls for suffix reservation and backward execution.

2 Related Work

2.1 Parameter-Efficient Fine-Tuning

PEFT methods differ in the parameters they train. Adapters introduce small modules between pretrained layers (Houlsby et al., 2019). Prefix tuning optimizes continuous prefix activations (Li & Liang, 2021), while prompt tuning learns soft input prompts (Lester et al., 2021). IA^3 rescales activations with learned vectors (Liu et al., 2022). LoRA trains low-rank weight updates (Hu et al., 2022), and QLoRA combines these updates with a frozen 4-bit NormalFloat (NF4) backbone, double quantization and paged optimizers (Dettmers et al., 2023). These approaches make task-specific adaptation more compact, although activation costs still depend on the training inputs.

Activation-efficient variants address this remaining storage. LoRA-FA freezes the down-projection factor, avoiding storage of its full input activations (Zhang et al., 2023). CARE-LoRA reconstructs inputs from low-rank features while retaining both trainable factors (Zhang et al., 2026). LoRAct uses sampled low-rank activation compression (Shi et al., 2025). CoSelTune combines such compression with suffix reservation and selective document backpropagation.

2.2 Long-Context Fine-Tuning

Context-extension methods adapt models beyond their pretrained sequence lengths. LongLoRA combines low-rank adaptation with shifted sparse training attention (Chen et al., 2024). LongQLoRA adds quantization and position interpolation (Yang, 2023), while YaRN modifies rotary position scaling for efficient context extension (Peng et al., 2024). CoSelTune targets adaptation cost under context settings shared by the compared methods.

Execution and token selection offer complementary savings. FlashAttention-2 improves exact attention through tiling and work partitioning (Dao, 2024). SeCO constructs and differentiates one chunk's graph at a time, and SpaCO selectively backpropagates through chunks (Li et al., 2025). TokenTune instead selects individual positions uniformly (Simoulin et al., 2024). TokenSeek replaces random selection with attention- and gradient-based importance scores (Zeng et al., 2026). Their whole-sequence selection includes supervised prediction positions in our comparison (Appendix F.1). CoSelTune reserves the complete suffix within that budget and uses text-based document selection. Its execution module coordinates both regions through causal scans, shared storage and recomputation to improve overall resource use.

3 Background

The input consists of a document $\mathcal{D}$ followed by a question-and-answer suffix. We preserve full-context forward computation and answer-token cross-entropy (CE) loss $\mathcal{L}$. The suffix is the region reserved for state backpropagation, while only answer targets contribute to the loss.

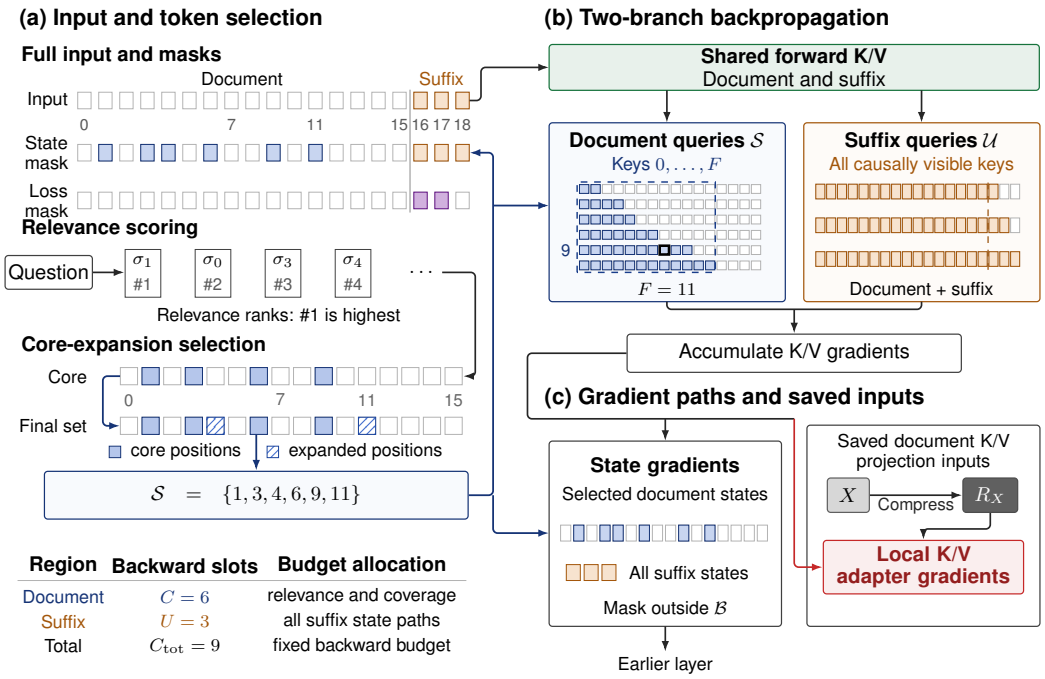


Figure 1: **Overview of CoSelTune.** (a) We first rank document blocks σ_i by question relevance, select *core* positions (solid blue) from highly ranked blocks, and add *expanded* positions (hatched blue) from a wider pool for broader coverage, forming the selected document set $\mathcal{S}$. (b) During backward, queries in $\mathcal{S}$ need keys only through $F = \max \mathcal{S}$, whereas the complete question-and-answer suffix requires full causal context. We therefore process these query groups separately over shared K/V. (c) We then sum their K/V gradients for local adapter updates. To reduce storage for these updates, we save document K/V projection inputs in compressed form. State gradients propagate to earlier layers only through $\mathcal{S}$ and the suffix. The full forward context and all answer losses are preserved.

Let D and U denote the numbers of tokens in the document and suffix, respectively, with positions $\mathcal{P} = \{0, \ldots, D-1\}$ and $\mathcal{U} = \{D, \ldots, D+U-1\}$. The document selector returns $\mathcal{S} \subseteq \mathcal{P}$, with count $1 \leq C = |\mathcal{S}| \leq D$ and frontier $F = \max \mathcal{S}$, the last selected position. Execution combines these positions with the suffix. Bars indicate gradients, hats reconstructed values, and tildes approximations. T denotes transpose. $\mathbb{R}$ denotes real numbers and I a matching identity matrix.

4 Methodology

4.1 Overview

CoSelTune reduces fine-tuning cost by restricting where hidden-state gradients propagate while retaining every token in the forward context. Given a document followed by a question-and-answer suffix, it computes the loss on all answer targets and reserves the entire suffix within a fixed budget for state backpropagation. A lightweight selector assigns the remaining budget to document positions using question relevance and coverage.

With the complete suffix retained, a common scan boundary stays at the sequence end. CoSelTune therefore separates query work while keeping shared key/value (K/V) tensors intact (Figure 1). Document queries stop at F, while suffix queries use all causally visible keys, including unselected document positions beyond F. These positions can still contribute to local K/V adapter updates, so we compress the projection inputs needed for those updates. Shared storage and recomputation reduce the remaining activation memory.

4.2 Token Selection

4.2.1 Suffix Reservation

We retain state-gradient paths for the entire question-and-answer suffix, while computing the loss only on answer targets. These are different masks: the state mask controls propagation to earlier layers, while the loss mask specifies supervised predictions. Losses use the original next-token alignment, including the position that predicts the first answer token.

Let $\eta \in (0, 1]$ be the token-selection budget ratio, specifying the fraction of input positions allocated to state backpropagation. The total capacity is C_{tot}. Reserving the U suffix positions leaves document capacity C and retained set $\mathcal{B}$:

$$C_{\text{tot}} = \lfloor \eta(D + U) \rfloor, \qquad C = C_{\text{tot}} - U, \qquad \mathcal{B} = \mathcal{S} \cup \mathcal{U}. \tag{1}$$

We require $U < C_{\text{tot}}$, leaving at least one document position. Table 3(a) evaluates this allocation. The selector fills the remaining budget with exactly C document positions.

4.2.2 Document Selection

Relevance scoring. To select document positions, we rank contiguous 256-word segments by BM25-style relevance to the question (Robertson & Zaragoza, 2009). Scores use document and question text alone, without a model-scoring pass. Appendix B gives the scoring formula, preprocessing and mapping to original token positions.

Core-expansion selection. We convert this ranking into exactly C positions. A core from highly ranked segments receives about two thirds of the budget. Expansion fills gaps within a wider ranked pool to obtain expanded positions while preserving the core.

The pools $\mathcal{V}_4, \mathcal{V}_8$ start with the top four and eight eligible segments, respectively. Each grows in rank order until it can supply C positions. The rule is

$$S_{\text{core}} = \text{Core}(\mathcal{V}_4, C), \qquad \mathcal{S} = \text{Expand}(S_{\text{core}}, \mathcal{V}_8, C), \qquad |\mathcal{S}| = C. \tag{2}$$

Core places evenly spaced anchors and fills gaps within segments. Expand fills the largest remaining gaps within segments in the wider pool. The pools can coincide when extended to meet the budget. All-zero relevance scores trigger global even spacing followed by gap filling. Appendix B specifies quotas and tie-breaking.

The output $\mathcal{S}$ contains original document positions and is reused across layers without shortening or reordering the input. Its last position $F = \max \mathcal{S}$ bounds document-query scans. The selector uses relevance and coverage, and execution exploits the resulting scan limit.

4.3 Two-Branch Backpropagation

State and parameter gradients. State gradients pass information to earlier layers, while local parameter gradients update the current layer's adapters. These gradients can involve different document positions. A retained query may use keys and values from an unselected position. That position can then contribute to the K/V adapter gradients, even though its own input-state gradient is stopped.

We therefore accumulate local parameter contributions before masking the state gradient returned to earlier layers. For layer inputs H, frozen weight W_0, rank-r down- and up-projection factors A, B and scale s, the forward projection is

$$Y = HW_0^{\mathsf{T}} + sZ_H B^{\mathsf{T}}, \qquad Z_H = HA^{\mathsf{T}}. \tag{3}$$

We first specify the local rule with uncompressed projection inputs. Section 4.4 introduces reconstruction for the local A gradient. Let $M_{\mathcal{B}}$ be the diagonal mask of retained positions, $\bar{Y}$ be the incoming output gradient and $G = s\bar{Y}$. Then

$$\bar{H} = M_{\mathcal{B}}\big(\bar{Y}W_0 + (GB)A\big), \qquad g_B = G^{\mathsf{T}} Z_H, \qquad g_A = (GB)^{\mathsf{T}} H. \tag{4}$$

The mask acts on $\bar{H}$, while local K/V parameter gradients include unselected rows used by retained queries. Locally, these inputs are held fixed while projection parameters remain differentiable

(Appendices A.4 and C.6). The implementation gathers incoming gradient rows at selected document positions before the input-gradient matrix products, computing only C document rows. Full K/V-gradient buffers remain for local adapter updates.

Shared K/V gradients. The retained states form two query groups with different causal requirements. Selected document queries need no keys beyond F, whereas suffix queries retain access to every causally visible key. Both groups use the same forward key and value tensors K, V, whose gradients accumulate before the local projection update:

$$\bar{K} = \bar{K}_s + \bar{K}_u, \qquad \bar{V} = \bar{V}_s + \bar{V}_u. \tag{5}$$

Subscripts s, u identify contributions from selected document queries and suffix queries. For a document key at $i > F$, the document branch contributes zero, but the suffix branch can still contribute. Thus the two branches divide query work while preserving their contributions to shared K/V.

Causal scan truncation. We reduce document-query attention work by scanning keys $0, \ldots, F$. For a selected query at position j, keys after j remain causally masked. For each head, the visible pairs and rectangular scan size satisfy

$$N_{\text{visible}}(\mathcal{S}) = \sum_{j \in \mathcal{S}} (j+1) \le N_{\text{scan}}(\mathcal{S}) = C(F+1) \le CD. \tag{6}$$

The selected count C controls query rows, while $\rho = (F+1)/D$ is the fraction of document keys scanned. Truncating this branch introduces no additional approximation under the following conditions.

Proposition 1 (Causal scan equivalence)**.** *For fixed forward tensors and selected document queries, scanning only keys $0, \ldots, F$ preserves that branch's outputs and returned backward gradients in exact arithmetic. Its K/V gradients are zero after F, where suffix queries can still contribute.*

Every selected query has $j \le F$, so causality makes all contributions from $i > F$ zero. Appendix C.2 gives the expanded proof.

4.4 Activation Management

Storage and recomputation. We retain full-context K/V shared by both query groups, attention outputs and row log-normalizers (log-sum-exp, LSE) for retained queries, and compressed document projection inputs for local K/V adapter updates. The retained-query statistics support recomputation of attention probabilities in small tiles (Dao, 2024), bounding the temporary score workspace. Selected-query and selected-document feed-forward network (FFN) activations are also recomputed. Appendix C.7 gives the tile sizes and derivatives.

FFN blocking. The full-document forward pass has a separate temporary-memory cost in the FFN. Since the FFN acts independently on token rows, we process document rows in blocks and release each block's expanded activations after writing its output. The output still contains every document position. Appendix C.8 gives the block size and rowwise equivalence.

Projection-input compression. Selective state propagation leaves document-wide dependencies in K/V adapter updates: suffix queries can use K/V from unselected document positions. We compress the projection inputs saved for these updates. The K/V tensors read by attention retain their forward values.

For each document K/V projection, $X = H_{\mathcal{P}} \in \mathbb{R}^{D \times d}$ contains the document rows of its input, with hidden width d. Following CARE-LoRA (Zhang et al., 2026), we retain $Z = XA^{\mathsf{T}}$ and fit $M \in \mathbb{R}^{r \times d}$ by ridge regression on the current forward activations. Let G be the scaled incoming gradient for these document rows, including both query branches' contributions. The saved representation and approximate A gradient are

$$\begin{aligned} R_X &= (Z, M), \qquad \widehat{X} = ZM, \\ \widetilde{g}_A &= (GB)^{\mathsf{T}} \widehat{X} = \left[(GB)^{\mathsf{T}} Z\right] M. \end{aligned} \tag{7}$$

We hold M fixed during backward. Evaluating the bracketed product first avoids constructing the $D \times d$ matrix $\widehat{X}$. At fixed G, the B gradient remains exact because it uses Z, while the local A error depends on $X - \widehat{X}$ and G. Appendices A.1 and C.3 give the reconstruction details, alternatives and error bound.

Algorithm 1 combines the selection, two query branches and storage rules for an L-layer model. The forward cache contains shared K/V, attention outputs, LSE, compressed projection inputs and states needed for recomputation. Query and FFN intermediates are reconstructed during backward. Appendix A.5 gives the detailed algorithm, including index alignment and optimizer operations.

Algorithm 1 One adapter update with selected backpropagation

Require: Document $\mathcal{D}$, question q, answer targets y, total budget C_{tot}, backbone and adapters
$C \leftarrow C_{\text{tot}} - |\mathcal{U}|$, $\mathcal{S} \leftarrow \textsc{CoreExpansionSelect}(\mathcal{D}, q, C)$
$\mathcal{B} \leftarrow \mathcal{S} \cup \mathcal{U}$, $F \leftarrow \max \mathcal{S}$
Run the full-context forward pass and save the layer caches
Compute the answer loss and initialize its backward gradient
for layers $l = L, L-1, \ldots, 1$ **do**
    Recompute the required local activations from the saved cache
    Backpropagate document queries in $\mathcal{S}$ over keys $0, \ldots, F$
    Backpropagate suffix queries over all causally visible keys
    Sum both branches' contributions to shared K/V gradients
    Accumulate local adapter gradients, using compressed inputs for document K/V
    Pass retained state-gradient rows in $\mathcal{B}$ to earlier layers
end for
Update the adapters with the accumulated parameter gradients

4.5 Analysis and Discussion

Computational cost. Text-based selection avoids a model-scoring pass. Its output determines the document branch's scan range. For one decoder layer at batch size one, let H_q be the number of query heads and d_h their width. Attention-backward work satisfies

$$\mathcal{W}_{\text{attn}} = O\left(H_q d_h \left[\underbrace{C(F+1)}_{\text{document queries}} + \underbrace{U(D+U)}_{\text{suffix queries}} \right] \right). \tag{8}$$

With query and key tile sizes B_q, B_k, the temporary score workspace is $O(H_q B_q B_k)$ elements. Compression stores $Dr + rd$ elements per document K/V input, a fraction $r/d + r/D$ of the full input. Shared K/V, full-context forward computation, parameter projections and optimizer work contribute separately to memory and time. A shorter document scan therefore reduces only part of the complete update cost. Section 5.3 analyzes backward execution and input compression. Section 5.4 examines suffix reservation, document selection, scan truncation and FFN blocking.

Approximation scope. Selective state propagation changes gradient paths relative to full backpropagation. For fixed forward tensors and selected queries, causal scan truncation preserves the document branch in exact arithmetic. Input reconstruction separately approximates the local A gradient at a fixed incoming gradient. Rowwise FFN blocking is also equivalent in exact arithmetic, though grouping can change bfloat16 (BF16) rounding. These rules apply to LoRA and QLoRA during training. Inference uses the standard full-context forward pass with the trained adapters.

5 Experiments

5.1 Experimental Setup

Training. We evaluate Qwen3-14B-Base (Yang et al., 2025) and Llama-3.1-8B (Grattafiori et al., 2024) with rank-4 Q/K/V adapters in 32-bit floating point (FP32), batch size one, a 32K window and AdamW at $2 \cdot 10^{-4}$ (Loshchilov & Hutter, 2019). Each model/precision/seed group shares eight warm-up updates before 128 LongAlpaca updates (Chen et al., 2024). Baselines are full-token LoRA (Hu

Table 1: **Quality and resource use.** Three-seed means except first-seed CARE/external baselines. Long-context costs use fixed inputs (Qwen3 64K, Llama 58K–64K). Bold: best fine-tuned result per group, excluding external baselines. Total/Step: seconds. Peak: GiB. GC+/GC−: checkpointing on/off. OOM: out of memory. Qwen3 32K: Appendix I.2.

	Quality ↑						Training ↓		Long context ↓	
Method	QASPER	SciMRC	Pirá	SciQA	arXiv	Avg.	Total	Peak	Step	Peak
Qwen3-14B										
Pretrained	43.65	38.90	57.26	15.95	26.55	36.46	–	–	–	–
Independent external PEFT baselines / BF16										
IA^3	44.36	41.94	57.53	16.51	27.58	37.59	1995.31	50.65	44.27	82.38
Bone	43.48	39.67	57.51	16.08	28.95	37.14	1953.46	46.10	43.71	71.54
LoRA / BF16										
$\text{LoRA}^{\text{GC+}}$	**47.03**	42.57	57.54	16.46	29.62	38.64	2259.37	46.37	79.88	72.17
LoRA + $\text{CARE}^{\text{GC+}}$	45.30	41.54	**58.40**	16.75	29.70	38.34	2325.36	46.13	81.02	71.55
LoRA + $\text{TokenTune}^{\text{GC−}}$	44.24	41.43	57.73	16.36	27.94	37.54	1578.24	56.71	OOM	OOM
LoRA + $\text{TokenTune}^{\text{GC+}}$	44.24	41.43	57.73	16.36	27.94	37.54	1999.11	**42.60**	74.37	63.23
LoRA + $\text{TokenSeek}^{\text{GC−}}$	45.31	37.55	56.79	**16.83**	**29.82**	37.26	2136.25	56.71	OOM	OOM
LoRA + $\text{TokenSeek}^{\text{GC+}}$	45.31	37.55	56.79	**16.83**	**29.82**	37.26	2562.77	**42.60**	74.25	63.23
LoRA + $\text{CoSelTune}^{\text{GC−}}$	46.91	**43.74**	57.40	16.61	29.78	**38.89**	**1443.82**	42.92	**53.28**	**59.59**
QLoRA / NF4										
$\text{QLoRA}^{\text{GC+}}$	**44.52**	40.54	57.66	16.64	28.98	**37.67**	2265.71	28.11	79.58	53.91
QLoRA + $\text{CARE}^{\text{GC+}}$	43.83	40.73	**57.83**	16.02	28.58	37.40	2333.03	27.87	80.82	53.29
QLoRA + $\text{TokenTune}^{\text{GC−}}$	44.21	**41.22**	56.37	15.75	27.61	37.03	1579.79	38.45	62.94	78.48
QLoRA + $\text{TokenTune}^{\text{GC+}}$	44.21	**41.22**	56.37	15.75	27.61	37.03	2014.21	**24.34**	74.53	44.96
QLoRA + $\text{TokenSeek}^{\text{GC−}}$	44.19	35.27	56.63	14.42	**29.23**	35.95	2141.34	38.45	62.82	78.48
QLoRA + $\text{TokenSeek}^{\text{GC+}}$	44.19	35.27	56.63	14.42	**29.23**	35.95	2576.45	**24.34**	74.37	44.96
QLoRA + $\text{CoSelTune}^{\text{GC−}}$	43.32	40.44	57.56	**17.03**	28.84	37.44	**1482.31**	24.81	**53.37**	**41.33**
Llama-3.1-8B / LoRA / BF16										
Pretrained	12.75	12.50	35.50	16.57	20.10	19.48	–	–	–	–
$\text{LoRA}^{\text{GC+}}$	**17.41**	**16.87**	46.13	**16.26**	**29.84**	**25.30**	1324.13	28.01	46.23	45.84
LoRA + $\text{TokenTune}^{\text{GC−}}$	14.40	13.07	**48.04**	15.39	23.07	22.79	**950.57**	33.67	37.60	59.30
LoRA + $\text{TokenTune}^{\text{GC+}}$	14.40	13.07	**48.04**	15.39	23.07	22.79	1184.64	25.46	43.99	39.79
LoRA + $\text{TokenSeek}^{\text{GC−}}$	12.80	12.35	32.81	8.74	24.79	18.30	1290.21	33.67	37.58	59.30
LoRA + $\text{TokenSeek}^{\text{GC+}}$	12.80	12.35	32.81	8.74	24.79	18.30	1522.60	25.46	43.97	39.79
LoRA + $\text{CoSelTune}^{\text{GC−}}$	17.37	16.29	45.50	16.24	29.45	24.97	953.67	**25.11**	**33.11**	**37.15**

et al., 2022), CARE (Zhang et al., 2026), TokenTune (Simoulin et al., 2024), TokenSeek (Zeng et al., 2026), IA^3 (Liu et al., 2022) and Bone (Kang, 2024). QLoRA (Dettmers et al., 2023) repeats the Qwen3 comparison with NF4 weights. Main selective comparisons use $\eta = 0.1$, a whole-sequence 10% budget including CoSelTune's complete suffix. Full-token, TokenTune, TokenSeek and CoSelTune scores are means over three initialization seeds (Appendix E). Ablations, CARE and external baselines use the first seed. TokenTune's sampling seed stays fixed.

Evaluation. Our 777 evaluation examples comprise LongBench's 200 QASPER questions (Bai et al., 2024), 150 answerable SciMRC questions (Zhang et al., 2024), all 227 Pirá English MRC test examples (Pirozelli et al., 2023), and samples of 100 ScientificQA test questions (Saikh et al., 2022) and 100 arXiv test papers (Cohan et al., 2018). Avg. equally weights four token-F1 scores and arXiv ROUGE-L on the 0–100 scale. All methods use greedy decoding within 32K.

Measurements. Total includes warm-up, 128 updates, selection/scoring and learning-rate trials, excluding evaluation and I/O. Primary training Total and per-run Peak average three seeds for each GC setting. Gradient checkpointing (GC) (Chen et al., 2016) rows share task scores (Appendix D). On one RTX PRO 6000, long-context Step averages three input medians of three restored updates each. Peak is maximum allocated GiB. Full-token references use GC, and CoSelTune uses internal recomputation.

5.2 Main Results

Qwen3-14B. CoSelTune achieves quality comparable to full-token LoRA/QLoRA while improving the mean over TokenTune and TokenSeek at the same 10% budget (Table 1). Its BF16 average is 38.89 versus LoRA's 38.64, gaining 1.35 and 1.63 points over TokenTune and TokenSeek. Its NF4 average is 37.44 versus QLoRA's 37.67, gaining 0.40 and 1.49 points over the two token baselines. Against GC-off TokenSeek, CoSelTune reduces Total by 32.4% in BF16 and 30.8% in NF4, and training peak by 24.3% and 35.5%.

GC-enabled token baselines use slightly less training memory than CoSelTune (42.60 versus 42.92 GiB in BF16, 24.34 versus 24.81 GiB in NF4). At 64K, CoSelTune lowers GC-enabled TokenSeek's step time from 74.25 to 53.28 seconds in BF16 and from 74.37 to 53.37 seconds in NF4, with lower peaks of 59.59 and 41.33 GiB.

Llama-3.1-8B. CoSelTune averages 24.97, comparable to full-token LoRA's 25.30 and above TokenTune's 22.79 and TokenSeek's 18.30. GC-off TokenTune has a slightly lower training Total, 950.57 versus 953.67 seconds. CoSelTune's long-context step takes 33.11 seconds at 37.15 GiB, the lowest among the compared methods.

5.3 Efficiency Analysis

Backward execution. We align document positions, the complete suffix and the 128-update protocol (Appendix F.2). Against TokenTune execution, CoSelTune lowers Total from 1520.34 to 1463.53 seconds (3.7%) and peak from 56.71 to 42.92 GiB (24.3%), with Avg. 38.57 versus 38.70. This evaluates the combined gradient routing, saved-input representation, blocking and recomputation.

Input compression. With the same document K/V projection-input representation, CoSelTune is 1.89–1.95× faster than full-state execution (Table 2). Within CoSelTune, CARE gives the fastest step (9.50 seconds) and lowers peak from 52.26 to 42.95 GiB against full saved inputs. CARE, 4-bit and sampled low-rank representations use less memory than their full-state counterparts, but 8-bit uses more (47.93 versus 46.77 GiB).

Table 2: **Input compression.** Matched document K/V projection-input representations on three documents (24,907–27,415 tokens). Full-state uses GC, and CoSelTune uses internal recomputation. Step follows Section 5.1. Peaks are allocated GiB.

Representation	Full Step (s)	Full Peak	CoSelTune Step (s)	CoSelTune Peak	Speedup
Full saved inputs	18.37	46.63	9.62	52.26	1.91×
CARE (default)	18.49	46.64	9.50	42.95	1.95×
8-bit activations	18.64	46.77	9.77	47.93	1.91×
4-bit activations	18.73	46.71	9.90	45.56	1.89×
Sampled low-rank (64)	18.62	46.64	9.73	43.21	1.91×

5.4 Ablation Studies

Suffix reservation. At the fixed budget $\lfloor 0.1(D+U) \rfloor$, reallocating about 90% of suffix slots to the document lowers Avg. by 0.73 on Qwen3 and 0.96 on Llama (Table 3(a)). More document states do not recover these losses.

We then test resource savings with complete suffixes for all methods on Qwen3, using $U + \lfloor 0.1D \rfloor$ positions. CoSelTune scores 38.82 versus 38.99 for TokenTune and 38.67 for TokenSeek (Table 3(b)), reducing Total by 9.3% and 32.9%, respectively, and peak memory by 25.9% against both.

Document selection. We replace the document selector while keeping backward execution and the complete suffix fixed (Table 4). Scores are close: 38.67 for core-expansion, 38.83 for TokenSeek and 38.57 for Global Random. Core-expansion has the lowest Total, 1396.53 seconds. Preparation takes 14.00 seconds versus TokenSeek's 562.60 seconds.

Table 3: **Suffix reservation.** BF16 LoRA. (a) Fixed total $\lfloor 0.1(D+U) \rfloor$, with suffix slots reassigned to the document. (b) Qwen3 with complete suffix plus 10% document positions, $U + \lfloor 0.1D \rfloor$, GC off. Total excludes warm-up. Times are seconds and Peak is GiB.

(a) Fixed-budget allocation.

Backbone	Suffix	Avg.	Total (s)	Peak (GiB)
Qwen3-14B	Complete	38.67	1396.53	42.92
Qwen3-14B	10%	37.94	1518.73	42.87
Llama-3.1-8B	Complete	24.28	925.49	25.11
Llama-3.1-8B	10%	23.33	1115.18	24.25

(b) Matched supervision.

Method	QASPER	SciMRC	Pirá	SciQA	arXiv	Avg.	Total (s)	Peak
CoSelTune	46.67	43.56	57.42	17.06	29.40	38.82	1465.76	44.19
TokenTune	47.10	43.98	57.29	17.38	29.20	38.99	1615.72	59.64
TokenSeek	46.67	42.90	56.95	16.97	29.86	38.67	2184.01	59.64

Table 4: **Document selection.** Fixed CoSelTune execution and full suffix. Prep: selection/scoring. Total: Prep plus 128 updates, excluding warm-up. Time in seconds, Peak in GiB.

Selector	Avg.	Prep	Total	Peak
Core-expansion	38.67	14.00	1396.53	42.92
TokenSeek	38.83	562.60	2019.80	42.92
Global Random	38.57	0.09	1463.53	42.92

Scan truncation. Truncating the document-query scan at F with positions fixed saves 23.1% and 12.4% of complete-step time at 32K and 64K (Table 5(a)). Separately, on 128 training inputs, mean scan fractions are 89.26%, 97.49% and 99.97% for CoSelTune, TokenSeek and Global Random under the same execution (Appendix B.4).

FFN blocking. We disable document FFN blocking while fixing inputs, selected positions, scans and compression. Enabling blocking saves 2.23 GiB at 32K and 4.48 GiB at 64K, adding 0.16 and 0.06 seconds per step, respectively (Table 5(b)).

Table 5: **Execution ablations.** Fixed inputs, positions and full suffix. (a) Full scans document keys, while To F stops at F. Saved: complete-step time reduction. (b) FFN blocking on/off at fixed scans and CARE inputs. Step averages three input medians of three restored updates each. Peak: maximum allocated GiB.

(a) Scan truncation.

Window	Full (s)	To F (s)	Saved (%)
32K	17.12	13.16	23.1
64K	60.80	53.28	12.4

(b) FFN blocking.

Window	FFN blocks	Step (s)	Peak (GiB)
32K	On	13.16	44.32
32K	Off	13.00	46.55
64K	On	53.28	59.59
64K	Off	53.22	64.07

6 Conclusion

CoSelTune reduces long-document training costs by reserving the supervised suffix and selecting document positions for relevance and coverage. Its backward pass uses these positions to limit attention scans and saved activations while retaining the full forward context. Experiments with LoRA and QLoRA show lower training costs with competitive task scores. At a fixed token budget, both backbones benefit from preserving the complete suffix rather than assigning more positions to the document.

REPRODUCIBILITY STATEMENT

The main text and appendices describe the method, derivations, experimental settings, per-seed results and resource measurement protocols. The supplementary archive provides implementation sources, data-preparation and training entry points, configurations, fixed sample identifiers, the recorded execution environment and results reported in the paper. The README describes dependencies and local checks; model weights and dataset text are obtained separately.

AI USE STATEMENT

Generative AI tools assisted with research-method discussion, dataset preparation and reformatting, experimental implementation and debugging, formulation and checking of mathematical arguments, interpretation of saved results, figure preparation, translation, and manuscript drafting. We used existing datasets without synthesizing data or modifying their original labels. Numerical tables and plots are computed from the recorded experiments. The authors are responsible for the final content of the manuscript and its associated artifacts.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, 2024. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. URL https://arxiv.org/abs/1604.06174.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongLoRA: Efficient fine-tuning of long-context large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.12307.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 615–621. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-2097. URL https://aclanthology.org/N18-2097/.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.08691.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115, 2023. doi: 10.52202/075280-0441. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract.html.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799, 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, 2018. doi: 10.18653/v1/P18-1031. URL https://aclanthology.org/P18-1031/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Jiale Kang. Bone: Block affine transformation as parameter efficient fine-tuning methods for large language models, 2024. URL https://arxiv.org/abs/2409.15371v1.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, 2021. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243/.

Wenhao Li, Yuxin Zhang, Gen Luo, Daohai Yu, and Rongrong Ji. Training long-context LLMs efficiently via chunk-wise optimization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2691–2700, 2025. doi: 10.18653/v1/2025.findings-acl.138. URL https://aclanthology.org/2025.findings-acl.138/.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A. Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1950–1965, 2022. doi: 10.52202/068431-0142. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/0cde695b83bd186c1fd456302888454c-Abstract-Conference.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022. doi: 10.52202/068431-2011. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.00071.

Paulo Pirozelli, Marcos M. José, Igor Silveira, Flávio Nakasato, Sarajane M. Peres, Anarosa A. F. Brandão, Anna H. R. Costa, and Fabio G. Cozman. Benchmarks for Pirá 2.0, a reading comprehension dataset about the ocean, the brazilian coast, and climate change. *arXiv preprint arXiv:2309.10945*, 2023. URL https://arxiv.org/abs/2309.10945.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Tanik Saikh, Tirthankar Ghosal, Amish Mittal, Asif Ekbal, and Pushpak Bhattacharyya. ScienceQA: a novel resource for question answering on scholarly articles. *International Journal on Digital Libraries*, 23(3):289–301, 2022. doi: 10.1007/s00799-022-00329-y. URL https://doi.org/10.1007/s00799-022-00329-y.

Jiang-Xin Shi, Wen-Da Wei, Jin-Fei Qi, Xuanyu Chen, Tong Wei, and Yu-Feng Li. Memory-efficient fine-tuning via low-rank activation compression, 2025. URL https://arxiv.org/abs/2509.23472.

Antoine Simoulin, Namyong Park, Xiaoyi Liu, and Grey Yang. Memory-efficient fine-tuning of transformers via token selection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 21565–21580, 2024. doi: 10.18653/v1/2024.emnlp-main.1202. URL https://aclanthology.org/2024.emnlp-main.1202/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Jianxin Yang. LongQLoRA: Efficient and effective method to extend context length of large language models. *arXiv preprint arXiv:2311.04879*, 2023. URL https://arxiv.org/abs/2311.04879.

Runjia Zeng, Qifan Wang, Qiang Guan, Ruixiang Tang, Lifu Huang, Zhenting Wang, Xueling Zhang, Cheng Han, and Dongfang Liu. TokenSeek: Memory efficient fine tuning via instance-aware token ditching. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2601.19739.

Gengyu Zhang, Haiyin Ran, Zhengbao He, Yuhang Liu, Hanling Tian, Zhehao Huang, and Xiaolin Huang. CARE-LoRA: Compressed activation REconstruction for memory-efficient LoRA. *arXiv preprint arXiv:2607.11940*, 2026. URL https://arxiv.org/abs/2607.11940.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. LoRA-FA: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303v1*, 2023. URL https://arxiv.org/abs/2308.03303v1.

Xiao Zhang, Heqi Zheng, Yuxiang Nie, Heyan Huang, and Xian-Ling Mao. SciMRC: Multi-perspective scientific machine reading comprehension. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 14418–14428. ELRA and ICCL, 2024. URL https://aclanthology.org/2024.lrec-main.1257/.

A Gradient Properties and Memory Analysis

This section explains how selected state gradients coexist with contributions from the full document, and how activation storage depends on these choices.

A.1 K/V Adapter Gradients with Compressed Activations

For each document K/V projection, let $X \in \mathbb{R}^{D\times d}$ contain the input activations and $Z = XA^\mathsf{T} \in \mathbb{R}^{D\times r}$ their low-rank features. Following CARE-LoRA (Zhang et al., 2026), we fit a reconstruction matrix $M \in \mathbb{R}^{r\times d}$ to these forward activations by ridge regression. The matrix maps each row of Z back to an approximation of the corresponding input row and satisfies

$$(Z^\mathsf{T}Z + \lambda I)M = Z^\mathsf{T}X, \qquad \widehat{X} = ZM. \tag{9}$$

Here $\lambda = 10^{-6}$ is the regularization strength and I is the $r \times r$ identity. We solve the system using FP32 statistics, retain $R_X = (Z, M)$ and hold M fixed during backward. Appendix C.3 derives this system from the reconstruction objective.

During backward, the incoming gradient combines contributions from selected document queries and suffix queries. With $\bar{Y} = \partial\mathcal{L}/\partial Y$ and $G = s\bar{Y}$, the document K/V adapter gradients are

$$g_B = G^\mathsf{T}Z, \qquad \widetilde{g}_A = (GB)^\mathsf{T}\widehat{X} = \big[(GB)^\mathsf{T}Z\big]M. \tag{10}$$

The product $(GB)^\mathsf{T}Z$ is only $r \times r$, so we evaluate it first and then multiply by M. This computes the approximate A gradient without constructing the $D \times d$ matrix $\widehat{X}$. The B gradient uses the retained forward features and is exact for the given incoming gradient.

These parameter gradients include document positions used by either attention branch, including unselected positions reached by suffix queries. Equation 4 separately restricts input gradients to the selected document positions. Reconstruction changes the local A gradient through the residual $X - \widehat{X}$. Equation 26 gives its exact error identity and norm bound for a fixed incoming gradient.

The same interface supports other reconstructions of X. We also evaluate 8-bit and 4-bit quantization and sampled low-rank compression (Shi et al., 2025), retaining the original Z for the B gradient in each case. These alternatives change the saved-input representation used for the A gradient while preserving token selection and the input-gradient rule. Appendix I.4 gives their settings.

A.2 Forward and Backward Paths within a Layer

The forward pass processes all D document positions. Differentiable state paths are retained for the selected document positions and complete suffix. Document query gradients follow the selected positions. Selected document queries and suffix queries share the document K/V tensors. Their loss gradients, also called adjoints, add before computing input-state and parameter gradients separately.

The CoSelTune graph retains upstream paths for selected document states and the complete suffix. Frozen tokenwise normalization has a block-diagonal Jacobian over positions, so masking its returned input gradient is equivalent in exact arithmetic to masking the corresponding projection-input gradient. Parameter gradients are accumulated separately. Applying the mask to K/V projection outputs would instead remove the suffix contributions that the method retains.

A schematic example is $D = 8$, $\mathcal{S} = \{1, 3\}$, and suffix positions $\{8, 9\}$. The branch for selected document queries scans keys 0:3. Position 6 has no active state path, but a suffix query can consume its K/V. Its B contribution is $G_6^\mathsf{T}Z_6$ and its reconstructed A contribution is $(G_6B)^\mathsf{T}\widehat{X}_6$. The input vector–Jacobian product (VJP) at position 6 remains zero.

A.3 Activation Storage

Table 6 summarizes the main activation terms for one layer at batch size one. H_q, H_{kv} are the query and K/V head counts, and d_h is the head width. Saved states and temporary workspaces have different lifetimes, so their sizes do not add directly to the measured peak. Model weights, optimizer states and other buffers also contribute.

Table 6: Principal activation-storage terms. Sizes count elements. Shared K/V tensors are counted once across the two attention branches.

Component	Size	Role
Selected document states	$C \times d$ per state	State backpropagation
Complete suffix states	$U \times d$ per state	Full answer supervision
Shared document K/V	$2H_{kv}Dd_h$	Context for both query branches
Compressed inputs Z, M	$Dr + rd$	Document K/V adapter gradients
Attention score tile	$H_q B_q B_k$	Temporary attention probabilities
FFN intermediates	$O(B_f d_{\text{ff}})$	Temporary expanded activations

The document forward output still contains Dd elements. Sharing avoids duplicate K/V storage, recomputation reduces saved intermediate states, and compression replaces the full-width inputs used by parameter gradients. Appendix C relates these terms to peak memory over a complete update.

A.4 Local Projection Derivatives

Define the backward rule. The selected-backpropagation rule retains state paths for selected document positions and the complete suffix. Its local derivatives separate state propagation from adapter updates. For input $X \in \mathbb{R}^{D\times d}$, frozen weight $W_0 \in \mathbb{R}^{d_{\text{out}}\times d}$ and low-rank factors $A \in \mathbb{R}^{r\times d}$, $B \in \mathbb{R}^{d_{\text{out}}\times r}$ used in LoRA and QLoRA, the forward computation is

$$Y = XW_0^{\mathsf{T}} + sZB^{\mathsf{T}}, \qquad Z = XA^{\mathsf{T}}. \tag{11}$$

Here d, d_{out} are the input and output widths, r is the adapter rank and s is the scale. Tokens form the rows of X and Y. The intermediate $Z \in \mathbb{R}^{D\times r}$ contains low-rank features, and $Y \in \mathbb{R}^{D\times d_{\text{out}}}$ is the output. In QLoRA, W_0 denotes the dequantized frozen weight used in computation. We omit layer indices.

Mask state gradients. Let $\bar{Y}^S$ be the incoming gradient with respect to the projection output Y, and let $G^S = s\bar{Y}^S$. Superscript S identifies the backward rule. For document row i, the returned input gradient is

$$\bar{X}_i^S = \begin{cases} \bar{Y}_i^S W_0 + (G_i^S B)A, & i \in \mathcal{S}, \\ 0, & i \in \mathcal{P} \setminus \mathcal{S}. \end{cases} \tag{12}$$

Selected rows propagate gradients to earlier computations, including the frozen projection term $\bar{Y}_i^S W_0$. Unselected rows keep their forward values but return zero input gradients. Appendix C.6 derives this rule by holding those input rows fixed during local differentiation.

The corresponding local K/V parameter derivatives are

$$g_B^S = (G^S)^{\mathsf{T}} Z, \qquad g_A^S = (G^S B)^{\mathsf{T}} X. \tag{13}$$

The sums include document rows used by suffix attention even when their input-state gradients are zero. These are local derivatives under the selected-backpropagation rule. Its incoming G^S can differ from full-state backpropagation. Query gradients follow the execution set $\mathcal{B}$, comprising selected document rows and the full suffix (Appendix C.3).

A.5 Detailed Training Algorithm

Algorithm 2 combines these operations into one update. The L-layer backbone has frozen weights $\mathcal{W}_0$ and adapters θ, with layer blocks θ_l. Ω stores AdamW moments and the step count. Superscript $+$ denotes the updated state.

The forward pass returns final states H_L and layer caches $\mathcal{C}_l$ containing recomputation states, K/V, attention outputs, LSE and compressed inputs. During backward, $\bar{H}$, $\bar{O}$ and R denote state, attention-output and residual gradients, and g_θ collects adapter gradients g_{θ_l}.

Algorithm 2 Detailed adapter update with selected backpropagation

Input: Document $\mathcal{D}$, question q, targets y, frozen L-layer backbone $\mathcal{W}_0$, adapters θ, optimizer state Ω
Input: Total budget C_{tot}, FFN block size $B_f = 1024$, document/suffix positions $\mathcal{P}, \mathcal{U}$
Output: Updated adapters θ^+ and optimizer state Ω^+

$C \leftarrow C_{\text{tot}} - |\mathcal{U}|$
$\mathcal{S} \leftarrow \textsc{CoreExpansionSelect}(\mathcal{D}, q, C)$
$\mathcal{B} \leftarrow \mathcal{S} \cup \mathcal{U} \quad F \leftarrow \max \mathcal{S}$
$(\mathcal{L}, H_L, \mathcal{C}) \leftarrow \textsc{FullForwardAndLoss}(\mathcal{D}, q, y, \mathcal{W}_0, \theta, \mathcal{B}, B_f)$
$\bar{H} \leftarrow \nabla_{H_L} \mathcal{L} \quad g_\theta \leftarrow 0$
for $l = L, L-1, \ldots, 1$ **do**
  $(\bar{O}, R) \leftarrow \textsc{RecomputeLocalBackward}(\bar{H}, \mathcal{C}_l, \mathcal{S}, \mathcal{U})$
  $(\bar{Q}_s, \bar{K}_s, \bar{V}_s) \leftarrow \textsc{SelectedAttentionBackward}(\bar{O}_{\mathcal{S}}, \mathcal{C}_l, 0{:}F)$
  $(\bar{Q}_u, \bar{K}_u, \bar{V}_u) \leftarrow \textsc{SuffixAttentionBackward}(\bar{O}_{\mathcal{U}}, \mathcal{C}_l, \mathcal{P} \cup \mathcal{U})$
  $(\bar{K}_s, \bar{V}_s) \leftarrow \textsc{ZeroPadKeys}(\bar{K}_s, \bar{V}_s, 0{:}F, \mathcal{P} \cup \mathcal{U})$
  $\bar{Q} \leftarrow \textsc{ScatterAdd}(\bar{Q}_s, \mathcal{S}, \bar{Q}_u, \mathcal{U})$
  $\bar{K} \leftarrow \bar{K}_s + \bar{K}_u \quad \bar{V} \leftarrow \bar{V}_s + \bar{V}_u$
  $(g_{\theta_l}, \bar{H}) \leftarrow \textsc{ProjectionBackward}(\bar{Q}, \bar{K}, \bar{V}, R, \mathcal{C}_l)$
  $g_\theta[\theta_l] \leftarrow g_{\theta_l} \quad \bar{H}_{\mathcal{P} \setminus \mathcal{S}} \leftarrow 0$
end for
$g_\theta \leftarrow \textsc{ClipGlobalNorm}(g_\theta, 1)$
$(\theta^+, \Omega^+) \leftarrow \textsc{AdamW}(\theta, \Omega, g_\theta)$
return (θ^+, Ω^+)

At each layer, the attention branches return gradients over different key ranges. ZEROPADKEYS aligns these ranges and SCATTERADD restores query positions. PROJECTIONBACKWARD then handles normalization, rotary embeddings and residual paths, using compressed inputs for document K/V adapter gradients (Appendix A.1). The state mask specifies the returned support. Document input gradients are computed on gathered rows, while K/V adjoints can be dense. We clip the accumulated parameter gradients at global norm one and update the adapters with AdamW.

B Selection Rule and Scan Distributions

B.1 Text Scoring and Token Membership

The selector receives the question, document text, and tokenizer character offsets. Long documents are partitioned into contiguous blocks of 256 whitespace-delimited words for scoring, preserving the original model input. Question and document terms are lowercased, restricted to alphanumeric strings and filtered for common English function words. The remaining question terms form a set. For document collections, title and body terms contribute to each document's counts, with title occurrences weighted by two.

Let $f_{t\sigma}$ be the weighted count of term t in segment σ, $\ell_\sigma = \sum_t f_{t\sigma}$, and $\bar{\ell}$ the mean segment length. With N segments and within-input document frequency n_t, the BM25-style score (Robertson & Zaragoza, 2009) is

$$\text{score}(\sigma, \mathcal{Q}) = \sum_{t \in \mathcal{Q}: f_{t\sigma} > 0} \log\left(1 + \frac{N - n_t + 0.5}{n_t + 0.5}\right) \frac{f_{t\sigma}(k_1 + 1)}{f_{t\sigma} + k_1(1 - b + b\ell_\sigma/\bar{\ell})}, \tag{14}$$

Here $\mathcal{Q}$ denotes the preprocessed question-term set, n_t counts segments containing t, and $k_1 = 1.2$, $b = 0.75$. Empty segments score zero. Ties follow title and segment order. Tokenizer offsets map segment spans to positions in the original input.

B.2 Core-Expansion Selection Rule

The document capacity is $C = C_{\text{tot}} - U$, where $C_{\text{tot}} = \lfloor \eta(D + U) \rfloor$ for the token-selection budget ratio η defined in Section 4.2.1. The execution module reserves the complete suffix before selection. The high-relevance pool starts with the top four eligible segments, and the coverage pool starts with the top eight. Each pool extends along the same ranking until its token capacity reaches C. Each pool is sized for the final budget before anchors are allocated. The pools can coincide when extra segments are needed.

Define the anchor and core counts as

$$C_a = \max(1, \lfloor C/3 \rfloor), \qquad C_c = \max(C_a, \lfloor 2C/3 \rfloor). \tag{15}$$

For the token pools $\mathcal{V}_4, \mathcal{V}_8$ defined in the main text, the full construction is

$$\begin{aligned} S_{\text{anchor}} &= \text{Even}(\mathcal{V}_4, C_a), \\ S_{\text{core}} &= \text{Expand}(S_{\text{anchor}}, \mathcal{V}_4, C_c), \\ \mathcal{S} &= \text{Expand}(S_{\text{core}}, \mathcal{V}_8, C). \end{aligned} \tag{16}$$

The first two lines define $\text{Core}(\mathcal{V}_4, C)$ in Equation 2. The coverage set is the final output. The nested sets satisfy $S_{\text{anchor}} \subseteq S_{\text{core}} \subseteq \mathcal{S}$.

Even distributes quotas across the selected segments in ranked round-robin order, skipping segments at capacity. For n ordered token members and quota $k > 1$, indices $\lfloor i(n-1)/(k-1) \rfloor$ for $i = 0, \ldots, k-1$ give evenly spaced positions. Quota one selects the midpoint. Expand preserves existing positions and adds quotas in the same order. A segment with no retained positions starts at its midpoint. Each subsequent position maximizes distance to the retained members, considering interior-gap midpoints and segment endpoints. Distances use member indices, with ties favoring the lower index. If all lexical scores are zero, the construction uses all document positions as one pool, with evenly spaced anchors and deterministic gap filling.

Algorithm 3 Core-expansion selection from ranked document segments

Require: Question, document segments, token offsets, document capacity C

Rank segments by lexical score and determine ordered token members
if all lexical scores are zero **then**
    Set $\mathcal{V}_4 = \mathcal{V}_8 = \mathcal{P}$, treating all document positions as one pool
else
    Build $\mathcal{V}_4, \mathcal{V}_8$, extending each ranked pool until its capacity reaches C
end if
$C_a \leftarrow \max(1, \lfloor C/3 \rfloor)$ $C_c \leftarrow \max(C_a, \lfloor 2C/3 \rfloor)$
$S_{\text{anchor}} \leftarrow \text{Even}(\mathcal{V}_4, C_a)$
$S_{\text{core}} \leftarrow \text{Expand}(S_{\text{anchor}}, \mathcal{V}_4, C_c)$
$\mathcal{S} \leftarrow \text{Expand}(S_{\text{core}}, \mathcal{V}_8, C)$
return sorted $\mathcal{S}$, $C = |\mathcal{S}|$, $F = \max \mathcal{S}$

B.3 Controls and Comparison Roles

The selector controls preserve the suffix and sample exactly $C = C_{\text{tot}} - U$ document positions. Global Random uses one fixed sample without replacement (seed 734003). TokenSeek ranks document positions by its warm-up-model scores. Both keep CoSelTune execution fixed. The main-table TokenTune and TokenSeek baselines instead select C_{tot} positions across the complete sequence. The scan control holds the complete selected set fixed and varies the key range.

B.4 Selected-Query Scan Ranges at a 10% Budget

Table 7 summarizes $100(F+1)/D$ from the selected positions on all 128 training inputs. The mean scan fraction is 89.26% for CoSelTune, 97.49% for TokenSeek and 99.97% for Global Random. The selected count is identical across methods for each input. These fractions describe the selected-query branch under CoSelTune execution, connecting position choice to its key-scan range.

Figure 2 shows the complete empirical distributions. At each horizontal value, the curve gives the fraction of training inputs whose frontier fraction is at most that value. The right panel expands the upper tail, where Global Random approaches a full document scan. These distributions use the same inputs and suffix-preserving 10% allocation as Table 4.

Table 7: Selected-query scan fractions on the training inputs. Values are percentages of document keys. P10/P90 are the 10th/90th percentiles, using linear interpolation. Each row summarizes all 128 inputs at the same 10% token budget.

Selection	Mean (%)	Median (%)	P10 (%)	P90 (%)
CoSelTune	89.26	93.70	73.97	99.83
TokenSeek	97.49	97.65	95.18	99.37
Global Random	99.97	99.98	99.93	100.00

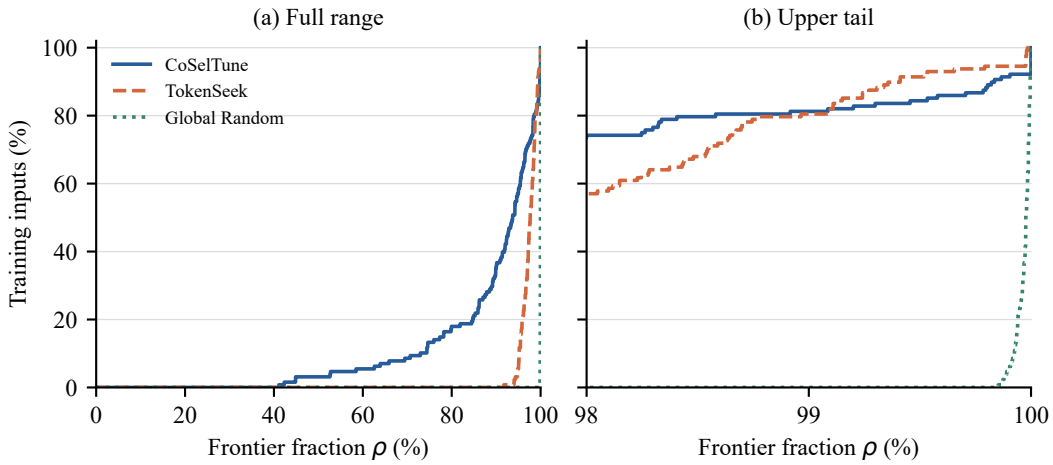


Figure 2: Complete frontier-fraction distributions under the whole-sequence 10% budget with complete suffix reservation. Each curve includes all 128 training inputs. The vertical axis is the percentage of inputs with frontier fraction at most ρ. Panel (b) enlarges the 98–100% range.

C Extended Resource and Local-Gradient Derivations

For matrices of the same shape E_1, E_2, use $\langle E_1, E_2 \rangle_F = \mathrm{tr}(E_1^\mathsf{T} E_2)$, where tr is the trace. We use $\|\cdot\|_F$ for the Frobenius norm, and $\|\cdot\|_2$ for the spectral norm of matrices or Euclidean norm of vectors. Identity matrices match the surrounding dimensions.

C.1 Uniform Frontier and Execution Cost

Proposition 2 (Last position under uniform selection)**.** *For a uniform size-C subset of $\{0, \ldots, D-1\}$ sampled without replacement, $1 \le C \le D$, and an integer m counting leading document positions,*

$$\Pr(F + 1 \le m) = \frac{\binom{m}{C}}{\binom{D}{C}} \le (m/D)^C, \quad C \le m \le D, \tag{17}$$

$$\mathbb{E}[F + 1] = \frac{C(D + 1)}{C + 1}. \tag{18}$$

Proof. There are $\binom{m}{C}$ subsets inside the first m positions, and each factor $(m - j)/(D - j)$ is at most m/D. The $D - C$ omitted positions form $C + 1$ exchangeable gaps, so the mean gap after the last selected position is $(D - C)/(C + 1)$. □

Proposition 2 applies to uniform subsets without replacement. The expected unscanned tail is $(D - C)/(C + 1)$, giving

$$\frac{\mathbb{E}[F + 1]}{D} = 1 - \frac{D - C}{D(C + 1)}. \tag{19}$$

This connects position selection to the implementation's rectangular scan bound. At fixed C, the position rule can change F. The full-scan comparison in Table 5(a) fixes the selected set and varies the key range.

For one query head, work for selected document queries is bounded by $C(F + 1)d_h$ up to constant factors. The complete suffix requires at most $O(U(D + U)d_h)$ attention work. We decompose the update time as

$$t_{\text{update}} = t_{\text{forward}} + t_{\text{selected}}(C, F) + t_{\text{suffix}}(U, D) + t_{\text{parameter}} + t_{\text{optimizer}}. \tag{20}$$

The terms represent non-overlapping parts of an update, separating scan-dependent work from other costs. Timings from overlapping profiling regions cannot be added.

C.2 Attention Adjoints from the Two Query Branches

Proof of Proposition 1.

Proof. A selected query satisfies $j \le F$. Every removed key has $i > F \ge j$, so its masked softmax probability is zero. Removing it changes neither normalization nor derivatives. The suffix branch continues to use all document keys. □

Equation 5 separates the shared K/V gradients by their query source. For a key after F, selected-document queries contribute zero because the causal mask excludes it. Suffix queries can still contribute. Appendix C.7 derives each branch from the masked softmax and shows how to accumulate it over key tiles. Grouped-query attention sums query-head contributions into the shared K/V heads. Rotary and tokenwise normalization backward preserve token support when mapping to raw projection outputs.

C.3 Projection Differentials and Retained Information

For one document projection, $\bar{Y} = \partial \mathcal{L}/\partial Y$ and $G = s\bar{Y}$. Differentiating with X fixed gives

$$\mathrm{d}Y = s\, X A^\mathsf{T} \mathrm{d}B^\mathsf{T} + s\, X \mathrm{d}A^\mathsf{T} B^\mathsf{T}. \tag{21}$$

Applying $\mathrm{d}\mathcal{L} = \langle \bar{Y}, \mathrm{d}Y \rangle_F$ and separately differentiating with respect to X gives the local gradients

$$\bar{X} = \bar{Y}W_0 + (GB)A, \qquad g_B = G^\mathsf{T}Z, \qquad g_A = (GB)^\mathsf{T}X. \tag{22}$$

Here $\bar{X} = \partial\mathcal{L}/\partial X$, $g_A = \partial\mathcal{L}/\partial A$ and $g_B = \partial\mathcal{L}/\partial B$. The frozen branch contributes to the input derivative. Masking the input gradient gives Equation 4, while parameter gradients still sum over all document rows. The custom backward therefore gives the two gradients different token supports.

CARE-LoRA's reconstruction (Zhang et al., 2026) minimizes

$$J(M) = \|X - ZM\|_F^2 + \lambda \|M\|_F^2 . \tag{23}$$

The first term measures reconstruction error, while the second regularizes the reconstruction coefficients. Differentiating with respect to M gives

$$\nabla_M J = 2Z^\mathsf{T}(ZM - X) + 2\lambda M = 0, \tag{24}$$

which yields the normal equation in Equation 9. For $\lambda > 0$ and nonzero u, the quadratic form $\|Zu\|_2^2 + \lambda \|u\|_2^2$ is positive, giving a unique solution. CoSelTune retains Z, M and evaluates

$$\widetilde{g}_A = (GB)^\mathsf{T}ZM = [Z^\mathsf{T}(GB)]^\mathsf{T}M. \tag{25}$$

Factorized evaluation avoids materializing $\widehat{X} = ZM$ as a saved full-width backward activation. Construction uses FP32 statistics and $\lambda = 10^{-6}$. M is held fixed in the reconstruction backward rule.

The retained statistics use $Dr + rd$ elements, a fraction $r/d + r/D$ of the full input. Their precision determines the byte count. Computation includes low-rank projections, reconstruction statistics and factor-gradient accumulation. Shared K/V tensors contribute the separate storage term in Table 6.

At fixed forward activations and incoming $G = s\bar{Y}$, the A-gradient reconstruction error satisfies

$$g_A - \widetilde{g}_A = (GB)^\mathsf{T}(X - \widehat{X}), \qquad \|g_A - \widetilde{g}_A\|_F \le \|GB\|_2 \left\|X - \widehat{X}\right\|_F . \tag{26}$$

Equation 26 also holds for any reconstructed input used only in this local A map, including quantized or independently factorized inputs. At fixed state support and numerical forward, these replacements leave the input-state and B-gradient rules unchanged. This bound describes reconstruction at a fixed incoming gradient. The local objective below explains the separate effect of restricting state paths.

C.4 Uniform Frontier Moments

For $F_+ = F + 1$, the mass function is

$$\Pr(F_+ = m) = \frac{\binom{m-1}{C-1}}{\binom{D}{C}}, \qquad C \le m \le D. \tag{27}$$

Summing $m\binom{m-1}{C-1} = C\binom{m}{C}$ and $m(m+1)\binom{m-1}{C-1} = C(C+1)\binom{m+1}{C+1}$ gives

$$\mathrm{Var}(F_+) = \frac{C(D+1)(D-C)}{(C+1)^2(C+2)}. \tag{28}$$

The tail bound in Proposition 2 also implies $\Pr(\rho \le a) \le a^C$ for $0 \le a \le 1$, using $m = \lfloor aD \rfloor$. If $m < C$ the probability is zero. These identities apply to uniform fixed-size selection.

C.5 From Scan Volume to a Complete-Step Cost Model

For a fixed input and selected set, write a local execution model as

$$t(\ell_{\mathrm{scan}}) = a + bC\ell_{\mathrm{scan}}, \qquad a \ge 0,\ b > 0, \tag{29}$$

where ℓ_{scan} is the number of scanned document keys, a collects work outside its scan-dependent part, and b models cost per scanned query–key pair. Under this linear model, define $\alpha = bCD/(a + bCD)$. Then

$$1 - \frac{t(F+1)}{t(D)} = \alpha(1-\rho), \qquad \frac{t(D)}{t(F+1)} = \frac{1}{1 - \alpha(1-\rho)}. \tag{30}$$

The operation count is exact for the rectangular scan. The timing model's constants depend on kernel launches, tiling and other implementation costs. It predicts a smaller complete-step response when the selected scan is a smaller fraction of total work.

Changing $\mathcal{S}$ changes both visible-pair structure and execution geometry. Fixed-$\mathcal{S}$ scan truncation instead isolates the scan limit, as measured in Table 5(a). Equation 30 explains why the complete-step saving depends on the fraction of work spent on that scan.

C.6 Local Objective for Selected State Paths

At common parameters θ_t, freeze the unselected input rows at each interface prescribed by the CoSelTune backward rule. Let $H_e(\theta)$ be an incoming activation, H_e^t its common-state value, and $M_e \in \mathbb{R}^{n_e \times n_e}$ the diagonal 0/1 mask retaining selected document positions and all suffix positions, where $H_e \in \mathbb{R}^{n_e \times d_e}$. Here e indexes interfaces, n_e, d_e are their row counts and widths, and I is the matching identity. The interface becomes

$$U_e(\theta) = M_e H_e(\theta) + (I - M_e) H_e^t, \qquad \mathrm{d}U_e = M_e \, \mathrm{d}H_e. \tag{31}$$

The differential implies $\bar{H}_e = M_e \bar{U}_e$ because the mask is symmetric. This substitution preserves the forward value at θ_t while restricting activation gradients. The consuming projection retains its local parameter derivative, including document K/V contributions. Let $\widetilde{\mathcal{L}}_S$ denote the answer loss with these substitutions at the specified interfaces and the remaining parameter-dependent operations retained. With full-input local parameter gradients and exact arithmetic,

$$g^{S,\mathrm{exact}} = \nabla_\theta \widetilde{\mathcal{L}}_S(\theta, \theta_t)\Big|_{\theta_t}, \qquad \widetilde{\mathcal{L}}_S(\theta_t, \theta_t) = \mathcal{L}(\theta_t). \tag{32}$$

Because the fixed features are refreshed at every training step, this identity applies locally at the current parameters. It does not guarantee a decrease in the next-step training loss or validation loss. Input reconstruction introduces the separate local approximation in Equation 10.

C.7 Reusing Forward Statistics for a Streamed Attention VJP

For one attention head of width d_h, let $q_j, k_i, v_i \in \mathbb{R}^{d_h}$ denote query, key and value vectors, and let p_{ji} be the causally masked probability. For retained query j, save its forward output $o_j = \sum_{i \le j} p_{ji} v_i$ and row log-normalizer $\ell_j = \log \sum_{i \le j} \exp(q_j^\mathsf{T} k_i / \sqrt{d_h})$. Given output gradient $\bar{o}_j$, for a causal key tile $\mathcal{I}$, reconstruct p_{ji} and the scaled-logit gradient a_{ji} as

$$p_{ji} = \mathbf{1}_{\{i \le j\}} \exp(q_j^\mathsf{T} k_i / \sqrt{d_h} - \ell_j), \qquad a_{ji} = p_{ji} \left[\bar{o}_j^\mathsf{T} v_i - \bar{o}_j^\mathsf{T} o_j \right]. \tag{33}$$

The indicator $\mathbf{1}_{\{i \le j\}}$ is one for causally visible keys and zero otherwise. The term $-\bar{o}_j^\mathsf{T} o_j$ accounts for softmax normalization. With no attention dropout, the complete VJP follows by accumulating key-tile contributions and summing over retained queries $j \in \mathcal{S} \cup \mathcal{U}$:

$$\bar{q}_j = \frac{1}{\sqrt{d_h}} \sum_{\mathcal{I}} \sum_{i \in \mathcal{I}} a_{ji} k_i, \qquad \bar{k}_i = \frac{1}{\sqrt{d_h}} \sum_j a_{ji} q_j, \qquad \bar{v}_i = \sum_j p_{ji} \bar{o}_j. \tag{34}$$

The stored o_j already contains the global probability-weighted value sum, and ℓ_j supplies the global normalization. These standard attention-adjoint identities (Dao, 2024) therefore admit a pass over key blocks for the selected-query branch without materializing its entire probability matrix. CoSelTune gathers these statistics from the document forward pass and adds the resulting K/V adjoints to those from suffix queries.

For H_q query heads and fixed query/key tiles B_q, B_k, one score tensor contains at most $H_q B_q B_k$ elements. CoSelTune uses $B_q = 64$ and $B_k = 2048$: at $H_q = 40$, one FP32 score tile occupies 20 MiB. The evaluated TokenTune implementation processes one K/V head group at a time with query chunk 256 and the full key axis. With group size 5 and 64K total tokens, its corresponding FP32 probability tensor occupies 320 MiB. These individual-tensor sizes quantify the workspace mechanism. The GPU peak also includes temporary tensors, K/V and adjoint buffers, and saved states that are live at the same time.

Recomputation limits retained activations. Key tiling bounds the selected-attention probability workspace. The complete-update measurements in Table 1 include these buffers together with the remaining model state.

C.8 Document FFN Blocks and the Peak over Time

For the frozen, tokenwise gated FFN, let $W_g, W_u \in \mathbb{R}^{d_{\mathrm{ff}} \times d}$ and $W_d \in \mathbb{R}^{d \times d_{\mathrm{ff}}}$ be its gate, up and down weights. $\odot$ is elementwise multiplication and SiLU is the sigmoid linear unit. Write

$$\mathrm{FFN}(X) = \left[\mathrm{SiLU}(X W_g^\mathsf{T}) \odot (X W_u^\mathsf{T}) \right] W_d^\mathsf{T}. \tag{35}$$

For any partition of the token rows into blocks $\mathcal{I}_b$, row independence gives

$$\mathrm{FFN}(X)_{\mathcal{I}_b} = \mathrm{FFN}(X_{\mathcal{I}_b}), \qquad |\mathcal{I}_b| \le B_f. \tag{36}$$

This equality holds in exact arithmetic. CoSelTune evaluates the numerical document in blocks of at most $B_f = 1024$ rows and writes each result into its assigned slice of a preallocated $D \times d$ output. Each block's expanded-width intermediates are released before the next block. The expanded-width workspace is therefore $O(B_f d_{\mathrm{ff}})$, with the Dd output and normalization buffers accounted separately. The blocks use the same tokenwise FFN formula. The row dimensions of the matrix multiplications affect floating-point rounding.

Let $\mathcal{A}_{\mathrm{live}}(\tau)$ be the set of live backing allocations at time τ, counting shared views once. The allocated peak is

$$M_{\mathrm{peak}} = \max_{\tau} \sum_{b \in \mathcal{A}_{\mathrm{live}}(\tau)} \mathrm{bytes}(b). \tag{37}$$

This lifetime accounting connects saved states, shared K/V, numerical-forward buffers, and temporary tiles. A smaller retained-activation footprint can shift the peak to a different operation, so peak memory is measured over the complete update. The CoSelTune blocking on/off comparison in Table 5(b) separately measures the complete-step memory effect of this module.

D Experimental Conditions

D.1 Training Data and Optimization

Training uses LongAlpaca-12k (Chen et al., 2024) with a fixed instruction/response template. The document-end marker separates the document from the question, and each answer ends with an end-of-sequence (EOS) token. Answers contain at most 512 target tokens, and the question suffix is limited to 1024 tokens. Document groups are defined by normalized source filenames, or by context when filenames are unavailable. A deterministic hash partition assigns groups to training, validation and test subsets in an 80/10/10 split. Group-wise sampling fixes eight warm-up examples within a 16K window and 128 training examples within a 32K window. Documents average 25,360.086 tokens and range from 24,587 to 27,415.

The common configuration uses rank-4 Q/K/V adapters with scale 2, batch size one, and AdamW with constant learning rate $2 \cdot 10^{-4}$, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01 and global gradient clipping at 1 (Loshchilov & Hutter, 2019). Backbone weights and forward activations use BF16. Adapter parameters, gradients, optimizer moments, reconstruction statistics and attention-gradient accumulation use FP32. Attention dropout is disabled. Within each model/precision group and initialization seed, methods share the eight-update initialization, training order and optimizer settings.

Validation uses 53 held-out examples containing 13,690 answer targets. Training and warm-up document groups and exact document-token matches are excluded. The subset retains the first example of each remaining document group and distinct document-token sequence. Validation CE selects hyperparameters separately from the five-task evaluation.

D.2 Token Budgets and Experimental Controls

TokenTune, TokenSeek and CoSelTune use $C_{\mathrm{tot}} = \lfloor \eta(D+U) \rfloor$ backward positions with budget ratio $\eta = 0.1$. TokenTune and TokenSeek select across the complete sequence, including their required boundary positions in this capacity. Their losses use the original next-token labels at selected prediction positions. CoSelTune reserves the complete suffix and selects $C_{\mathrm{tot}} - U$ document positions. Full-token LoRA retains every position.

The primary comparison uses three initialization seeds for each model/precision group: 961800, 710167783 and 1740061989 for BF16 LoRA on both backbones, and 20260917, 710167783 and 1740061989 for QLoRA. Full-token tuning, TokenTune, TokenSeek and CoSelTune are trained and evaluated at each seed. Main-table quality equally averages their three scores on each task; Avg. then equally weights the five tasks. Appendix E gives the individual results. The pretrained model is evaluated once per backbone.

The first seed (961800 for BF16, 20260917 for NF4) is the reference for CARE, independent external baselines, ablations and the 128–256-update comparison. TokenTune and random-selection controls use the fixed sampling seed 734003, distinct from adapter initialization. Thus the three-seed comparison varies initialization while holding TokenTune's sampled positions fixed. TokenSeek scores each input at the corresponding shared warm-up checkpoint. Matched-position execution controls additionally share document positions and complete suffix supervision.

Gradient checkpointing recomputes activations during backward. Resource costs are measured separately with GC on and off. TokenTune and TokenSeek GC-on rows reuse the corresponding GC-off task scores, assuming checkpointing preserves forward computation and random number generator (RNG) state up to floating-point effects. CoSelTune uses internal selected-query and selected-FFN recomputation with whole-layer GC off.

Total training time includes warm-up, 128 updates, selection or scoring, and optimization of additional learning-rate trials. Evaluation, model loading and checkpoint I/O are excluded. Selection is timed on prepared inputs and charged once per run, including when scores are cached. The same boundary applies to all methods. For full-token tuning, TokenTune, TokenSeek and CoSelTune, training Total and Peak are means over three runs for each GC setting; Peak averages their individual maxima. CARE and external baselines also retain their reference-seed results. Long-context costs use the fixed-input measurements in Appendix I, separately from initialization-seed averages.

D.3 Independent Parameter-Efficient Baselines

IA^3 (Liu et al., 2022) trains FP32 scaling vectors on K/V outputs and FFN down-projection inputs, totaling 778,240 parameters. Bone (Kang, 2024) uses rank 12 on Q/K/V projections with 3,440,640 trainable FP32 parameters. Both use the same BF16 Qwen3-14B backbone and full-token backpropagation, with independent initialization and eight warm-up updates.

Each learning-rate trial is evaluated after 32 training updates. Validation selects 10^{-3} over $2 \cdot 10^{-4}$ for IA^3 (CE 1.213533 versus 1.255898), and $2 \cdot 10^{-4}$ over 10^{-3} for Bone (1.190003 versus 1.325215). The selected model completes 128 training updates. IA^3 reaches final validation CE 1.163151. Reported training totals include optimization for both learning-rate trials, counting shared updates once. Both baselines use GC in the long-context resource tests.

D.4 Activation-Compression Baseline

LoRA + CARE uses CARE-LoRA reconstruction (Zhang et al., 2026) with full-token state backpropagation and layer checkpointing. It shares the rank-4 Q/K/V adapters, eight-update initialization and 128 training inputs with the BF16 LoRA comparison. Its 32K/64K resource tests compress all-token Q/K/V inputs. Table 2 separately controls compression at the matched document K/V scope, allowing the execution effect to be assessed with the same saved-input representation.

D.5 Five-Task Evaluation

Table 8: The five evaluation datasets, generation budgets and metrics. Every method uses the same fixed examples.

Task	Examples	Maximum new tokens	Metric
QASPER	200	128	Token F1
SciMRC-Answerable	150	256	Token F1
Pirá 2.0 English MRC	227	128	Token F1
ScientificQA	100	128	Token F1
arXiv	100	512	ROUGE-L

QASPER uses LongBench's fixed 200-question cohort and prompt (Bai et al., 2024). SciMRC-Answerable uses 150 answerable questions from the authors' NLPCC 2023 train/development release (Zhang et al., 2024), comprising 102 extractive and 48 generative questions from 121 papers. They are reserved for evaluation here. Yes/no and unanswerable questions are excluded, and the subset is distinct from the benchmark's hidden test split.

Pirá 2.0 English MRC uses all 227 examples in the authors' test split (Pirozelli et al., 2023), with contexts from 181 scientific abstracts and 46 UN-report excerpts. ScientificQA uses a fixed sample of 100 questions from the 9,429-example author test split (Saikh et al., 2022), covering 93 abstracts. This is the scholarly-article ScienceQA dataset of Saikh et al. The arXiv summarization evaluation samples 100 papers uniformly without replacement from the 6,440-example test split (Cohan et al., 2018), using each paper's original abstract as its reference. ScientificQA and arXiv use sampling seed 20260921, with sample identities fixed before comparing methods.

Generation uses greedy decoding over the prepared prompt, a standard K/V cache and no chat template. The 32,768-token window reserves the generation budgets in Table 8. Excess-length prompts retain their beginning and end. Three arXiv prompts require truncation with Qwen3 and two with Llama, whose mean arXiv input lengths are 9,634 and 9,011 tokens. Token F1 evaluates the four QA datasets, and ROUGE-L evaluates arXiv summaries. Scores average equally over examples within each dataset, then Avg. equally weights the five dataset scores on the 0–100 scale.

All 777 examples are reserved for evaluation. Pirá and ScientificQA contain short supporting texts, while QASPER, SciMRC and arXiv provide document-level evaluations. An exact 128-word-span comparison between the 136 warm-up/training documents and QASPER evaluation contexts found no matches.

D.6 Suffix-Allocation Control

Table 3(a) fixes the BF16 initialization, 128 updates and total capacity C_{tot}. The control samples $M = \max\{1, \lfloor 0.1U \rfloor\}$ suffix positions uniformly with seed 734003 and assigns $C_{\mathrm{tot}} - M$ positions to the unchanged document selector. It inserts no additional answer positions. Every input retains at least seven supervised prediction positions.

CoSelTune assigns an average of 2307.17 positions to the document and 253.81 to the suffix. The control assigns 2536.05 and 24.93, with identical integer totals for each input. Both retain full-context forward computation, the same document K/V compression scope and suffix K/V parameter contributions. The control uses tiled backward and recomputation for its selected suffix queries. This comparison therefore evaluates the joint effect of token allocation, supervision coverage and the resulting execution under one initialization and one fixed suffix sample.

E Results by Initialization Seed

Tables 9–11 report all five task scores and training costs for the three initialization seeds. Each table contains Qwen3-14B BF16 LoRA, Qwen3-14B NF4 QLoRA and Llama-3.1-8B BF16 LoRA. All runs use the same 128 ordered training examples and 777 evaluation examples, with eight shared warm-up updates within each model/precision/seed group. TokenTune keeps sampling seed 734003 fixed across initialization seeds.

The primary quality and training-cost entries in Table 1 are arithmetic means across the three initializations. Each table includes both GC settings for TokenTune and TokenSeek; GC-on task scores reuse the corresponding GC-off results. Full-token methods use GC on, and CoSelTune uses internal recomputation with whole-layer GC off. Total includes warm-up, selection/scoring and training; Peak is each run's maximum allocated memory.

Table 9: First initialization: seed 961800 for BF16 LoRA and seed 20260917 for NF4 QLoRA. Quality uses the 0–100 scale; Total is seconds and Peak is GiB. This is also the reference initialization for the single-seed controls.

Method	QASPER	SciMRC	Pirá	SciQA	arXiv	Avg.	Total	Peak
Qwen3-14B / BF16, seed 961800								
LoRA$^{\mathrm{GC+}}$	47.61	41.80	57.85	16.51	29.83	38.72	2261.67	46.37
LoRA + TokenTune$^{\mathrm{GC-}}$	44.49	42.02	58.05	16.97	27.76	37.86	1577.84	56.71
LoRA + TokenTune$^{\mathrm{GC+}}$	44.49	42.02	58.05	16.97	27.76	37.86	1998.58	42.60
LoRA + TokenSeek$^{\mathrm{GC-}}$	45.82	36.82	57.37	17.40	30.24	37.53	2140.11	56.71
LoRA + TokenSeek$^{\mathrm{GC+}}$	45.82	36.82	57.37	17.40	30.24	37.53	2561.17	42.60
LoRA + CoSelTune$^{\mathrm{GC-}}$	46.29	42.98	57.31	16.96	29.81	38.67	1454.17	42.92
Qwen3-14B / NF4, seed 20260917								
QLoRA$^{\mathrm{GC+}}$	45.20	41.37	57.55	16.96	28.89	38.00	2268.37	28.11
QLoRA + TokenTune$^{\mathrm{GC-}}$	44.30	41.39	56.44	15.67	27.13	36.99	1579.50	38.45
QLoRA + TokenTune$^{\mathrm{GC+}}$	44.30	41.39	56.44	15.67	27.13	36.99	2014.46	24.34
QLoRA + TokenSeek$^{\mathrm{GC-}}$	44.84	36.90	56.56	14.40	29.84	36.51	2142.59	38.45
QLoRA + TokenSeek$^{\mathrm{GC+}}$	44.84	36.90	56.56	14.40	29.84	36.51	2577.26	24.34
QLoRA + CoSelTune$^{\mathrm{GC-}}$	43.11	41.56	57.33	17.08	28.95	37.61	1493.82	24.81
Llama-3.1-8B / BF16, seed 961800								
LoRA$^{\mathrm{GC+}}$	18.03	18.27	44.83	14.98	29.38	25.10	1330.28	28.01
LoRA + TokenTune$^{\mathrm{GC-}}$	14.53	12.96	48.25	16.05	22.26	22.81	952.05	33.67
LoRA + TokenTune$^{\mathrm{GC+}}$	14.53	12.96	48.25	16.05	22.26	22.81	1184.96	25.46
LoRA + TokenSeek$^{\mathrm{GC-}}$	12.72	12.37	34.01	8.37	24.51	18.40	1293.11	33.67
LoRA + TokenSeek$^{\mathrm{GC+}}$	12.72	12.37	34.01	8.37	24.51	18.40	1528.23	25.46
LoRA + CoSelTune$^{\mathrm{GC-}}$	16.85	16.02	44.17	14.97	29.41	24.28	957.74	25.11

Table 10: Second initialization: seed 710167783 for all model/precision groups. Metrics and units follow Table 9.

Method	QASPER	SciMRC	Pirá	SciQA	arXiv	Avg.	Total	Peak
Qwen3-14B / BF16, seed 710167783								
LoRA$^{\mathrm{GC+}}$	46.89	43.25	57.44	16.27	29.64	38.70	2257.59	46.37
LoRA + TokenTune$^{\mathrm{GC-}}$	43.87	41.76	57.73	15.90	27.97	37.45	1578.05	56.71
LoRA + TokenTune$^{\mathrm{GC+}}$	43.87	41.76	57.73	15.90	27.97	37.45	1999.81	42.60
LoRA + TokenSeek$^{\mathrm{GC-}}$	45.08	36.64	56.76	16.86	29.60	36.99	2133.41	56.71
LoRA + TokenSeek$^{\mathrm{GC+}}$	45.08	36.64	56.76	16.86	29.60	36.99	2571.85	42.60
LoRA + CoSelTune$^{\mathrm{GC-}}$	46.79	44.45	57.85	17.22	29.74	39.21	1441.70	42.92
Qwen3-14B / NF4, seed 710167783								
QLoRA$^{\mathrm{GC+}}$	44.52	39.59	57.58	16.59	28.79	37.41	2264.41	28.11
QLoRA + TokenTune$^{\mathrm{GC-}}$	44.29	39.88	56.41	15.76	27.91	36.85	1579.18	38.45
QLoRA + TokenTune$^{\mathrm{GC+}}$	44.29	39.88	56.41	15.76	27.91	36.85	2013.62	24.34
QLoRA + TokenSeek$^{\mathrm{GC-}}$	43.85	34.40	56.42	14.33	28.93	35.59	2140.75	38.45
QLoRA + TokenSeek$^{\mathrm{GC+}}$	43.85	34.40	56.42	14.33	28.93	35.59	2574.92	24.34
QLoRA + CoSelTune$^{\mathrm{GC-}}$	42.97	40.44	57.73	16.73	28.76	37.33	1478.55	24.81
Llama-3.1-8B / BF16, seed 710167783								
LoRA$^{\mathrm{GC+}}$	16.23	14.87	46.12	17.47	30.20	24.98	1320.79	28.01
LoRA + TokenTune$^{\mathrm{GC-}}$	14.22	13.25	48.73	14.84	23.93	22.99	949.56	33.67
LoRA + TokenTune$^{\mathrm{GC+}}$	14.22	13.25	48.73	14.84	23.93	22.99	1184.62	25.46
LoRA + TokenSeek$^{\mathrm{GC-}}$	12.43	12.47	34.04	9.44	25.17	18.71	1288.31	33.67
LoRA + TokenSeek$^{\mathrm{GC+}}$	12.43	12.47	34.04	9.44	25.17	18.71	1515.98	25.46
LoRA + CoSelTune$^{\mathrm{GC-}}$	17.77	17.87	45.92	17.01	29.24	25.56	959.17	25.11

Table 11: Third initialization: seed 1740061989 for all model/precision groups. Metrics and units follow Table 9.

Method	QASPER	SciMRC	Pirá	SciQA	arXiv	Avg.	Total	Peak
Qwen3-14B / BF16, seed 1740061989								
LoRA^{GC+}	46.58	42.65	57.34	16.61	29.41	38.52	2258.84	46.37
LoRA + TokenTune^{GC-}	44.36	40.51	57.42	16.20	28.09	37.32	1578.83	56.71
LoRA + TokenTune^{GC+}	44.36	40.51	57.42	16.20	28.09	37.32	1998.94	42.60
LoRA + TokenSeek^{GC-}	45.04	39.17	56.24	16.22	29.63	37.26	2135.24	56.71
LoRA + TokenSeek^{GC+}	45.04	39.17	56.24	16.22	29.63	37.26	2555.27	42.60
LoRA + CoSelTune^{GC-}	47.65	43.77	57.04	15.65	29.78	38.78	1435.58	42.92
Qwen3-14B / NF4, seed 1740061989								
QLoRA^{GC+}	43.83	40.66	57.84	16.38	29.26	37.59	2264.35	28.11
QLoRA + TokenTune^{GC-}	44.05	42.39	56.28	15.82	27.78	37.26	1580.69	38.45
QLoRA + TokenTune^{GC+}	44.05	42.39	56.28	15.82	27.78	37.26	2014.53	24.34
QLoRA + TokenSeek^{GC-}	43.89	34.52	56.91	14.52	28.90	35.75	2140.66	38.45
QLoRA + TokenSeek^{GC+}	43.89	34.52	56.91	14.52	28.90	35.75	2577.17	24.34
QLoRA + CoSelTune^{GC-}	43.86	39.32	57.62	17.26	28.82	37.37	1474.54	24.81
Llama-3.1-8B / BF16, seed 1740061989								
LoRA^{GC+}	17.97	17.48	47.43	16.35	29.95	25.84	1321.32	28.01
LoRA + TokenTune^{GC-}	14.46	13.00	47.14	15.29	23.01	22.58	950.11	33.67
LoRA + TokenTune^{GC+}	14.46	13.00	47.14	15.29	23.01	22.58	1184.35	25.46
LoRA + TokenSeek^{GC-}	13.26	12.20	30.40	8.42	24.68	17.79	1289.22	33.67
LoRA + TokenSeek^{GC+}	13.26	12.20	30.40	8.42	24.68	17.79	1523.61	25.46
LoRA + CoSelTune^{GC-}	17.49	14.97	46.40	16.73	29.70	25.06	944.11	25.11

F Additional Controlled Comparisons

F.1 Baseline Comparison Conditions

TokenTune and TokenSeek both retain the full forward context while restricting token backpropagation. TokenSeek combines *Instance-Aware Token Seeking* and *Efficient Token Ditching*. CoSelTune reserves the suffix, selects document positions from text and organizes backward computation around the resulting allocation. Table 12 summarizes the comparison conditions under the common Qwen3 protocol.

Table 12: Method differences under the evaluated protocol. Main baselines sample the full sequence. CoSelTune reserves the suffix. Fixed-position controls isolate the effect of backward execution.

Decision	TokenTune	TokenSeek	CoSelTune
Token allocation	Uniform whole-sequence sampling	Whole-sequence attention and gradient scoring	Reserved suffix, relevance and coverage in the document
State and parameter gradients	Selected-token backpropagation	Same token-selective training graph	Reserved suffix and selected document states, shared K/V accumulation
Saved inputs for K/V parameter gradients	Selected rows	Selected rows	Low-rank statistics for reconstructed A gradients
Preparatory computation	Sampling	One score pass at the warm-up checkpoint	Text-based selection

TokenTune uses exact-count global random sampling with one fixed sample (seed 734003). The first and last positions are included inside the total capacity. Remaining positions follow the uniform sampling policy. Selected prediction positions retain their original next-token labels.

TokenSeek combines last-layer head-mean attention column sums with a signed channel sum of the final decoder-input gradient. It applies gradient min–max normalization, an attention log transform and gradient weight 0.7. Scores are computed once at the shared warm-up checkpoint and held fixed. Both boundary positions count within C_{tot}. Preparation includes the scoring pass and constructing the selected set. GC-on/off resource measurements use the same positions and loss.

F.2 Execution with a Fixed Random Sample

The CoSelTune Random and matched TokenTune execution controls use identical document positions sampled with seed 734003 and preserve the complete suffix. Both start from the same BF16 LoRA warm-up checkpoint and follow the same 128 updates. This comparison evaluates the complete execution modules, including gradient routing, saved-input storage, blocking and recomputation.

Table 13: Execution controls with identical random positions and supervision. Each method uses one fixed sample. Total includes sampling and 128 updates, excluding the common warm-up. Peak is allocated GiB.

Execution	QASPER	SciMRC	Pirá	SciQA	arXiv	Avg.	Total (s)	Peak
CoSelTune	46.66	43.17	57.28	16.65	29.08	38.57	1463.53	42.92
TokenTune	46.81	42.36	57.93	16.80	29.60	38.70	1520.34	56.71

With positions and supervision fixed, CoSelTune averages 38.57 versus 38.70 for TokenTune execution. It reduces total compute from 1520.34 to 1463.53 seconds and peak memory from 56.71 to 42.92 GiB. The main TokenTune baseline instead samples across the complete sequence, including the supervised suffix.

F.3 Effect of Training Duration

We compare the same BF16 LoRA, TokenTune, TokenSeek and CoSelTune training trajectories at 128 and 256 updates, using reference initialization seed 961800 at both endpoints. The 128-update entries are this seed's results rather than the main-table means. All methods receive the next 128 unique LongAlpaca training examples, with batch size one, no gradient accumulation and the same constant learning rate $2 \cdot 10^{-4}$. The eight warm-up updates are separate. TokenTune retains sampling seed 734003. TokenSeek scores the additional examples at the original warm-up checkpoint. Evaluation uses the same 777 examples, prompts and decoding settings as at 128 updates.

Table 14: Training duration on Qwen3-14B-Base with BF16 LoRA. Δ Avg. is the 256-update average minus the 128-update average. Total includes warm-up, selection/scoring and training, in seconds. GC policies match the corresponding main-table rows.

Method	Updates	QASPER	SciMRC	Pirá	SciQA	arXiv	Avg.	Δ Avg.	Total (s)
LoRA$^{\text{GC+}}$	128	47.61	41.80	57.85	16.51	29.83	38.72	–	2261.67
LoRA$^{\text{GC+}}$	256	45.29	40.31	57.04	14.62	30.27	37.50	-1.22	4464.97
LoRA + TokenTune$^{\text{GC}-}$	128	44.49	42.02	58.05	16.97	27.76	37.86	–	1577.84
LoRA + TokenTune$^{\text{GC}-}$	256	44.77	38.66	58.27	15.55	28.76	37.20	-0.66	3116.10
LoRA + TokenSeek$^{\text{GC}-}$	128	45.82	36.82	57.37	17.40	30.24	37.53	–	2140.11
LoRA + TokenSeek$^{\text{GC}-}$	256	44.47	38.26	57.64	16.62	29.02	37.20	-0.33	4234.50
LoRA + CoSelTune$^{\text{GC}-}$	128	46.29	42.98	57.31	16.96	29.81	38.67	–	1454.17
LoRA + CoSelTune$^{\text{GC}-}$	256	44.40	39.80	56.44	15.44	30.52	37.32	-1.35	2832.48

Doubling the updates lowers the five-task average for all four methods: by 1.22 points for LoRA, 0.66 for TokenTune, 0.33 for TokenSeek and 1.35 for CoSelTune. CoSelTune's arXiv ROUGE-L improves from 29.81 to 30.52, while its four QA scores decline, reducing Avg. from 38.67 to 37.32 and increasing total compute from 1454.17 to 2832.48 seconds. Under this protocol, extending training to 256 updates increases cost and lowers the average score for all four methods.

F.4 Comparisons at the 10% Token Budget

Tables 1 and 4 distinguish whole-sequence baseline selection from document selector replacement at fixed CoSelTune execution. Both comparisons use the same total capacity per input. The selector controls also preserve the complete suffix, so their preparation and execution costs can be compared at a fixed allocation.

G Quantized-Backbone Evidence

The QLoRA comparison uses Qwen3-14B-Base with NF4 weights, double quantization and BF16 computation (Dettmers et al., 2023). Embeddings, the language-model head and normalization layers remain unquantized. Rank-4 Q/K/V adapters and AdamW states use FP32. At each initialization seed, the quantized variants share an eight-update warm-up and the same 128 training inputs. Evaluation uses the five cohorts totaling 777 examples, with individual seed results in Appendix E.

QLoRA and QLoRA + CARE use full-token backpropagation with GC. TokenTune and TokenSeek select the same whole-sequence 10% count, including prediction positions. CoSelTune reserves the complete suffix and applies core-expansion selection to the remaining document budget. GC-on/off costs are measured separately, with task-score reuse as specified in Appendix D.

CoSelTune uses 13.31 seconds and 26.07 GiB at 32K, and 53.37 seconds and 41.33 GiB at 64K. Its mean total training cost is 1482.31 seconds, compared with 2141.34 for GC-off TokenSeek. Its mean quality of 37.44 is comparable to full-token QLoRA's 37.67 and exceeds TokenTune's 37.03 and TokenSeek's 35.95. GC-on TokenTune averages 24.34 GiB during training, compared with CoSelTune's 24.81 GiB. Training peaks and long-context resource peaks describe different input sets and are reported separately.

H Cross-Model Evidence on Llama-3.1-8B

H.1 Model and Evaluation Conditions

Llama-3.1-8B (Grattafiori et al., 2024) uses the same eight warm-up and 128 LongAlpaca training examples as Qwen3-14B, with native tokenization and a 32,768-token training and evaluation window. Documents average 25,325 tokens. All methods are evaluated on the same five cohorts totaling 777 examples. The long-context resource inputs contain 65,536, 59,378 and 65,536 native Llama tokens and use the measurement protocol in Appendix I.

Llama has 32 layers, hidden width 4096, FFN width 14336, 32 query heads, 8 K/V heads and head dimension 128. Qwen3-14B has 40 layers, hidden width 5120, FFN width 17408, 40 query heads and 8 K/V heads, with the same head dimension. Rank-4 Q/K/V adapters contain 2,359,296 trainable parameters in Llama and 3,604,480 in Qwen3. Llama uses its native projections, root mean square normalization (RMSNorm) and rotary position embeddings. Its native context window supports up to 131,072 positions. Token positions are matched within each model family after tokenization.

H.2 Suffix Allocation

The suffix-allocation control fixes Llama's initialization seed 961800, 128 updates and total capacity C_{tot}. It samples 10% of suffix positions with seed 734003 and gives the remaining capacity to the unchanged document selector. Both variants retain full-context forward computation and the same document K/V compression scope.

Complete-suffix reservation averages 24.28 versus 23.33 for suffix sampling. It improves QASPER, SciMRC and arXiv while scoring lower on Pirá and ScientificQA. The gain is 0.96 points on average, consistent with the 0.73-point gain on Qwen3-14B.

H.3 Token-Selection Comparison

TokenTune and TokenSeek use the same three Llama initialization seeds, 128 ordered updates and five evaluation cohorts as CoSelTune (Appendix E). They use the whole-sequence budget

Table 15: Llama-3.1-8B suffix allocation at fixed total capacity. Each row uses one fixed selection configuration and the same five evaluation cohorts.

Suffix policy	QASPER	SciMRC	Pirá	SciQA	arXiv	Avg.
Complete suffix	16.85	16.02	44.17	14.97	29.41	24.28
10% suffix	13.58	14.01	47.08	17.16	24.79	23.33

$C_{\text{tot}} = \lfloor \eta(D+U) \rfloor$ with $\eta = 0.1$ and compute loss at selected supervised positions. TokenTune uses sampling seed 734003 and TokenSeek scores inputs at each shared warm-up initialization. The three long-context resource inputs are fixed. GC-on/off costs are measured separately, and their task scores follow the reuse policy in Appendix D.

Table 1 shows a three-seed average of 24.97 for CoSelTune, compared with 25.30 for full-token LoRA, 22.79 for TokenTune and 18.30 for TokenSeek. CoSelTune has the lowest long-context step time and peak memory among these token-selective methods. GC-off TokenTune has a slightly lower mean total training cost, 950.57 versus 953.67 seconds.

I Additional Resource Results

I.1 Measurement Conditions

Qwen3-14B training and the Llama-3.1-8B resource comparisons use an RTX PRO6000 Blackwell Server Edition with 94.971 GiB memory, a common 0.95 process allowance and a 600 W power limit. The software environment uses PyTorch 2.7.0+cu128, CUDA 12.8, Triton 3.6.0 and Transformers 5.16.1.

Timed updates include forward computation, supervised loss, backward, gradient clipping and the optimizer step. Loading, tokenization, checkpoint I/O and evaluation are excluded. Each long-context configuration uses three fixed inputs with one warm-up and three timed updates per input, starting from the same model and optimizer state. Step averages the per-input medians, and Peak is the maximum allocated memory across inputs. These resource tests use the first initialization. Training Total includes warm-up, 128 updates, selection/scoring and optimization of additional learning-rate trials. In contrast to the fixed-input resource tests, primary training Total and per-run Peak average three initialization seeds for each GC setting (Appendix D).

I.2 Qwen3-14B at 32K and 64K

Table 16 reports the five-task scores and resource measurements at 32,768 and 65,536 tokens. The resource inputs use YaRN factor 2 and the original supervised targets. Token-selective methods share the whole-sequence 10% capacity. BF16 LoRA variants share their eight-update initialization within each seed, while QLoRA, IA^3 and Bone use separate initializations under the same training-data protocol.

I.3 Per-Input Scan Effects

Table 17 shows the three document inputs separately. Each pair fixes the selected positions and changes only the key range. This isolates the effect of the scan boundary on complete-update time.

I.4 Compression at a Matched Scope

Table 2 compares full-state LoRA and CoSelTune with the same saved-input representation, initialization, optimizer and three documents of 24,907–27,415 tokens. Full-state LoRA uses layer checkpointing. CoSelTune uses its whole-sequence 10% capacity with the complete suffix and internal recomputation. Both compress document K/V inputs while retaining the original query and suffix calculations.

Table 16: Complete Qwen3-14B quality and resource measurements, including 32K and 64K. Averaging, units and GC policies follow Table 1. The 32K/64K columns use fixed-input measurements. GC-off full-token LoRA is OOM at both lengths. GC-off full-token QLoRA is OOM in the training-capacity test.

	Quality ↑						Training ↓		32K ↓		64K ↓	
Method	QASPER	SciMRC	Pirá	SciQA	arXiv	Avg.	Total	Peak	Step	Peak	Step	Peak
Pretrained	43.65	38.90	57.26	15.95	26.55	36.46	–	–	–	–	–	–
Independent external PEFT baselines / BF16												
IA^3	44.36	41.94	57.53	16.51	27.58	37.59	1995.31	50.65	16.25	54.98	44.27	82.38
Bone	43.48	39.67	57.51	16.08	28.95	37.14	1953.46	46.10	15.93	49.58	43.71	71.54
LoRA / BF16												
$\text{LoRA}^{\text{GC+}}$	**47.03**	42.57	57.54	16.46	29.62	38.64	2259.37	46.37	25.15	49.90	79.88	72.17
LoRA + $\text{CARE}^{\text{GC+}}$	45.30	41.54	**58.40**	16.75	29.70	38.34	2325.36	46.13	25.73	49.59	81.02	71.55
LoRA + $\text{TokenTune}^{\text{GC−}}$	44.24	41.43	57.73	16.36	27.94	37.54	1578.24	56.71	17.55	62.18	OOM	OOM
LoRA + $\text{TokenTune}^{\text{GC+}}$	44.24	41.43	57.73	16.36	27.94	37.54	1999.11	**42.60**	22.00	45.42	74.37	63.23
LoRA + $\text{TokenSeek}^{\text{GC−}}$	45.31	37.55	56.79	**16.83**	**29.82**	37.26	2136.25	56.71	17.55	62.18	OOM	OOM
LoRA + $\text{TokenSeek}^{\text{GC+}}$	45.31	37.55	56.79	**16.83**	**29.82**	37.26	2562.77	**42.60**	21.99	45.42	74.25	63.23
LoRA + $\text{CoSelTune}^{\text{GC−}}$	46.91	**43.74**	57.40	16.61	29.78	**38.89**	**1443.82**	42.92	**13.16**	**44.32**	**53.28**	**59.59**
QLoRA / NF4												
$\text{QLoRA}^{\text{GC+}}$	**44.52**	40.54	57.66	16.64	28.98	**37.67**	2265.71	28.11	25.10	31.63	79.58	53.91
QLoRA + $\text{CARE}^{\text{GC+}}$	43.83	40.73	**57.83**	16.02	28.58	37.40	2333.03	27.87	25.70	31.33	80.82	53.29
QLoRA + $\text{TokenTune}^{\text{GC−}}$	44.21	**41.22**	56.37	15.75	27.61	37.03	1579.79	38.45	17.67	43.92	62.94	78.48
QLoRA + $\text{TokenTune}^{\text{GC+}}$	44.21	**41.22**	56.37	15.75	27.61	37.03	2014.21	**24.34**	22.20	27.16	74.53	44.96
QLoRA + $\text{TokenSeek}^{\text{GC−}}$	44.19	35.27	56.63	14.42	**29.23**	35.95	2141.34	38.45	17.65	43.92	62.82	78.48
QLoRA + $\text{TokenSeek}^{\text{GC+}}$	44.19	35.27	56.63	14.42	**29.23**	35.95	2576.45	**24.34**	22.19	27.16	74.37	44.96
QLoRA + $\text{CoSelTune}^{\text{GC−}}$	43.32	40.44	57.56	**17.03**	28.84	37.44	**1482.31**	24.81	**13.31**	**26.07**	**53.37**	**41.33**

Table 17: Per-input scan comparisons at the 10% budget. Scan is $100(F+1)/D$. Full scans all document keys, while To F stops at the last selected position. Times are medians of three updates from the same initial state. Saved is the reduction in complete-update time.

Window	Doc.	D	C	$F+1$	Scan (%)	Full (s)	To F (s)	Saved (%)
32K	A	32446	2954	4971	15.32	16.89	8.01	52.6
32K	B	32541	3049	24981	76.77	17.24	14.94	13.4
32K	C	32543	3051	29917	91.93	17.23	16.53	4.1
64K	A	65214	6231	52739	80.87	59.71	51.52	13.7
64K	B	65309	6326	54764	83.85	61.21	53.98	11.8
64K	C	65311	6328	55248	84.59	61.48	54.33	11.6

The quantized representations use symmetric 64-feature groups, FP32 scales and nearest-even rounding. Sampled low-rank compression (Shi et al., 2025) uses rank 64, FP32 factors and no power iterations. The same per-layer sample is used across execution settings, and the adapter rank remains four. Both settings retain the original low-rank features for B gradients, so the comparison holds the compression rule fixed while changing backward execution.

I.5 FFN Blocking Control

Table 5(b) fixes the input, token selection, complete suffix, CARE compression, attention computation and optimizer. Blocking processes the document forward FFN in groups of at most 1024 token rows. The control processes all document rows together, keeping selected-gradient and suffix paths unchanged. Each length uses the same three-input measurement protocol. Row grouping can change BF16 rounding.

J TOKEN BUDGET COMPARISON

We compare whole-sequence budgets $\eta = 0.1$ and $\eta = 0.2$ on Qwen3-14B-Base with BF16 LoRA and NF4 QLoRA. Within each precision, both budgets use the same warm-up checkpoint, 128 training updates and 777 evaluation examples. Results use initialization seed 961800 for LoRA and 20260917 for QLoRA; TokenTune keeps sampling seed 734003. Whole-layer GC is off for all three methods, and CoSelTune retains its internal recomputation.

Table 18: **Token budget comparison.** Single-seed results under matched conditions. The 10% entries are the corresponding reference-seed results from Table 9. Quality and Avg. follow Table 1. Total includes warm-up, selection/scoring and 128 updates, in seconds; Peak is allocated training memory in GiB.

Method	η	QASPER	SciMRC	Pirá	SciQA	arXiv	Avg.	Total	Peak
Qwen3-14B / LoRA / BF16, seed 961800									
TokenTune	10%	44.49	42.02	58.05	16.97	27.76	37.86	1577.84	56.71
TokenTune	20%	46.12	41.06	57.52	16.51	28.22	37.89	2666.77	76.67
TokenSeek	10%	45.82	36.82	57.37	17.40	30.24	37.53	2140.11	56.71
TokenSeek	20%	46.17	38.53	55.86	16.60	30.08	37.45	3228.98	76.67
CoSelTune	10%	46.29	42.98	57.31	16.96	29.81	38.67	1454.17	42.92
CoSelTune	20%	46.68	42.96	58.06	16.13	29.72	38.71	2434.25	48.03
Qwen3-14B / QLoRA / NF4, seed 20260917									
TokenTune	10%	44.30	41.39	56.44	15.67	27.13	36.99	1579.50	38.45
TokenTune	20%	44.62	40.21	56.83	14.92	27.72	36.86	2660.84	58.41
TokenSeek	10%	44.84	36.90	56.56	14.40	29.84	36.51	2142.59	38.45
TokenSeek	20%	42.29	33.56	56.92	14.48	28.88	35.23	3223.61	58.41
CoSelTune	10%	43.11	41.56	57.33	17.08	28.95	37.61	1493.82	24.81
CoSelTune	20%	44.04	41.49	57.55	16.87	28.99	37.79	2441.24	29.80

At the 20% budget, CoSelTune achieves higher five-task average scores while using less training time and peak memory than both TokenTune and TokenSeek in both precision settings (Table 18).

For CoSelTune, increasing the budget from 10% to 20% improves Avg. by 0.04 points with LoRA and 0.18 points with QLoRA. Total increases by 67.4% and 63.4%, respectively, while Peak rises by 11.9% and 20.1%. These small quality gains at substantially higher resource cost give the 20% budget a less favorable overall balance than 10% in this setting.